

Nicholas Bodvin Sellevåg
Yan Senko
Fabian Kongelf
Oddbjørn S. Borge-Jensen

Exploring possibilities for GitLab as a Learning Management System

Bachelor's thesis in Digital Infrastructure and Cyber Security
Supervisor: Guoqiang Li
June 2022

Nicholas Bodvin Sellevåg
Yan Senko
Fabian Kongelf
Oddbjørn S. Borge-Jensen

Exploring possibilities for GitLab as a Learning Management System

Bachelor's thesis in Digital Infrastructure and Cyber Security
Supervisor: Guoqiang Li
June 2022

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Dept. of Information Security and Communication Technology

ABSTRACT

Title :	<u>Exploring possibilities for GitLab as a Learning Management System</u>	Date : 20/05/2022
Participants :	<u>Nicholas Bodvin Sellevåg</u> <u>Yan Senko</u> <u>Fabian Kongelf</u> <u>Oddbjørn S. Borge-Jensen</u>	
Supervisor(s) :	<u>Guoqiang Li</u>	
Employer :	<u>Erik Hjelmås</u>	
Keywords :	<u>Learning Management System, Automation, Web, Git, Programming (3-5)</u>	
Number of pages: 82	Number of appendix : 30	Availability : Open
Short description of the bachelor thesis :		
<p>The thesis is conducted as an experiment for exploring the possibilities of using Gitlab as a Learning Management System (LMS). There are several challenges an LMS should strive to solve. Erik Hjelmås, Associate Professor at Norwegian University of Science and Technology, has defined the challenges as follows:</p> <ol style="list-style-type: none">1. Provide easy access to public information2. Support User Access Control3. Publish announcements4. Handle assignments5. Perform digital testing6. Host discussion forums <p>Consequently, the group members of the thesis researched to which degree the challenges can be solved within the Git based system Gitlab. As a result, the group members have developed a system which demonstrates methods to solve the first four challenges defined by Erik Hjelmås. The developed system consists of several components to transform markdown documents into websites. Teachers can provide the system with their course material as markdown and automatically generate corresponding websites accessible for students. The GitLab solution allows users to circumvent the Graphical User Interface of generalized systems like BlackBoard Learn by using the Git command line interface (CLI) to change and upload files. Several of the professors the group interviewed for this thesis expressed significant enthusiasm for CLI-oriented methods in contrast to web interfaces.</p> <p>The group concludes that GitLab Pages is capable of offering a partially suitable replacement to the traditional LMS, but only for users with sufficient IT experience. There are however some things the existing systems do that GitLab cannot replicate. This includes the out of scope challenges of digital testing and discussion forums, but also some elegant solutions that requires a proper dedicated database, and not just a Git repository.</p>		

SAMMENDRAG

Tittel :	Exploring possibilities for GitLab as a Learning Management System	Dato : 20/05/2022
Deltaker(e) :	Nicholas Bodvin Sellevåg Yan Senko Fabian Kongelf Oddbjørn S. Borge-Jensen	
Veileder(e) :	Guoqiang Li	
Evt. oppdragsgiver :	Erik Hjelmås	
Stikkord/nøkkelord (3-5 stk) :	Learning Management System, Automation, Web, Git, Programming	
Antall sider/ord : 82	Antall vedlegg : 30	Publiseringsavtale inngått : Åpen
Kort beskrivelse av bacheloroppgaven : Bachelor oppgaven er utført som et eksperiment for å utforske muligheter ved å bruke Gitlab som en Læringsstyringsplattform (LMS). Det er en rekke utfordringer tilknyttet hva LMS tilstreber å tilby. Erik Hjelmås, Førsteamanuensis på Norges teknisk-naturvitenskapelige universitet, har definert utfordringene som følgende: <ol style="list-style-type: none">1. Gi enkel tilgang til offentlig informasjon2. Støtte brukertilgangskontroll3. Publiser kunngjøringer4. Håndtere innleveringer5. Utføre digital tester6. Tilgjengeliggjøre diskusjonsforum <p>Gruppen har utforsket til hvilken grad disse utfordringene kan løses gjennom bruk av GitLab Pages. Som et resultat av denne utforskningen har det blitt utviklet et system som demonstrerer metoder for å løse de første fire utfordringene som har blitt definert av Erik Hjelmås. Det utviklede systemet består av en rekke komponenter som lager en nettside basert på markdown filer. Forelesere kan laste opp sitt kursmateriale i systemet som markdown filer og disse vil automatisk brukes til å generere tilsvarende nettsider og gjort tilgjengelig for studentene. GitLab løsningen lar brukere unngå det grafiske brukergrensesnittet til generaliserte systemer som BlackBoard Learn ved å bruke Git terminalen for å endre og laste opp filer. Flere forelesere som gruppen har intervjuet for prosjektet har uttrykket entusiasme for en slik løsning ovenfor mer abstrakterte nett-grensesnitt.</p> <p>Gruppen har kommet frem til at Gitlab Pages kan tilby en delvis tilfredsstillende erstatning av en tradisjonell LMS, men kun spesifikt for IT-erfarne brukere. Det er derimot en rekke aspekter ved de eksisterende systemene som ikke kan gjenskapes i Gitlab. Dette inkluderer funksjonalitetene utenfor oppgavens omfang som for eksempel digital testing og tilgjengeliggjørelse av diskusjonsforum. Det er også flere elegante løsninger som påkrever dedikerte databaser og ikke bare et Git repository</p>		

Preface

This bachelor thesis concludes the bachelor degree program *Digital Infrastructure and Cyber Security* spanning three years at the Faculty of Information Technology and Electrical Engineering (Norwegian University of Science and Technology, NTNU). The thesis has been a collaborative project between four students; Nicholas Bodvin Sellevåg, Yan Senko, Fabian Kongelf and Oddbjørn S. Borge-Jensen. Group members have worked together on several projects throughout their studies at NTNU. The task description was given by Erik Hjelmås, Associate professor at the Department of Information Security and Communication Technology. Group members would like to address their gratitude towards Erik for expressing interest in the project, but foremost his role as our teacher and course coordinator.

Conducting the thesis would not be possible without the assistance of our supervisor Guoqi-ang Li. Weekly meetings discussing the purely academical aspects of our research process and thesis structure were a tremendous help throughout the project. We would also like to thank all professors and students who participated in our survey and took part in interviews.

Repositories

All code produced for the project is available at the following repositories. Only the working repository supports the GitLab Pages pipelines; the code repository was only made to have a redundant back-up repository on the git.gvk.idi GitLab instance.

Working repositories: <https://gitlab.stud.idi.ntnu.no/bachelor1>

Code repository: <https://git.gvk.idi.ntnu.no/Fabian/dcsg2900-gitlab-pages>

Table of Contents

List of Figures	viii
1 Introduction	1
1.1 Task	1
1.2 Problem area	2
1.3 Project scope	3
1.4 Demographic	4
1.5 Report Structure	5
2 Background	6
2.1 Field of study	6
2.2 Purpose	6
2.3 Motivation	7
2.4 Project Members	7
2.4.1 Organizing	8
2.4.2 Distribution of Workload	8
2.5 Experience	9
2.6 State of the art	9
2.6.1 Blackboard Learn	9
2.7 Related technologies	10
3 Methodology	15
3.1 System requirement analysis	16
3.1.1 Functionality to create website from markdown	16
3.1.2 Functionality to support access control	17
3.1.3 Functionality to publish announcements	17
3.1.4 Functionality to handle assignments	17
4 The developed system	17
4.1 System architecture	17
4.1.1 Architectural requirements	18
4.2 System Design	23

4.2.1	Information Publication via GitLab Pages	23
4.2.2	Exclusivity of private information with roles	25
4.2.3	Engaging announcements through automation	26
4.2.4	Deliverables	28
4.2.5	Confidence and integrity of software through testing	29
4.3	Page design	30
4.3.1	Visual Design	31
4.4	Implementation	40
4.4.1	Functionality to create website from markdown	40
4.4.2	Functionality to support access control	41
4.4.3	Functionality to publish announcements	44
4.4.4	Functionality to handle assignments	46
4.4.5	Docker-container	47
4.4.6	Timing-issues	48
4.4.7	Deploy stage in Gitlab CI	49
4.5	LMS installation and deployment	53
4.5.1	Prerequisites	55
4.6	File Structure Overview	55
4.7	Security	57
4.7.1	Endpoint Protection	57
4.7.2	GitLab's Security Measures	58
4.7.3	GitLab Access Control	62
4.7.4	Ruby	63
4.7.5	Jekyll	64
4.7.6	Container Security	65
4.7.7	Vulnerability scanning	66
4.7.8	Trusted Docker images	68
4.7.9	CI/CD best practices	70
4.8	Quality Assurance	70
4.8.1	Definition of Requirements	70
4.8.2	Automated Reports	71

5	Conclusion	73
5.1	Conclusion of possibilities for Gitlab as a Learning Management System	73
5.2	Evaluation	77
5.2.1	Organizing	77
5.2.2	Distribution of Workload	78
5.2.3	Project as a form of work	79
5.3	Reflection	79
5.3.1	Results	79
5.3.2	Discoveries	80
5.4	Critique of task	81
5.5	Future Considerations	82
A	Definition of Done	89
A.1	Functionality to create website from markdown	89
A.2	Functionality to support access control	92
A.3	Functionality to publish announcements	95
A.4	Functionality to handle assignments	97
B	Survey	99
C	Interviews	106
C.1	Mariusz Nowostawski	106
C.2	Ivar Farup	112
C.3	Frank Alexander Kramer	116
C.4	Rune Hjelsvold	119
C.5	Henrik Johnsen	122
D	Project Management	125
D.1	Time log	125
D.1.1	Nicholas Bodvin Sellevåg	125
D.1.2	Yan Senko	128
D.1.3	Fabian Kongelf	131
D.1.4	Oddbjørn S. Borge-Jensen	134
D.2	Project Planning	137

D.3	Gantt Chart	146
D.4	Project contract	148
D.5	Minutes of Meetings	155
D.5.1	Client meeting summaries	155
D.5.2	Guidance meeting summaries	160
D.6	Task	167

List of Figures

1	GitLab Pages flowchart	12
2	Container VS virtualization	15
3	Values of the architecture	19
4	Sequence diagrams for deliverables	23
5	Sequence Diagram of website generation	24
6	GitLab project hierarchy access levels	25
7	GitLab project hierarchy creation	26
8	Flowchart of five independent Docker Containers	27
9	Sequence Diagram of the "Notification" pipeline	27
10	Code snippet from sendAnnouncement script depicting the curl request	28
11	How the announcement looks on teams	29
12	The LMS's semantic element design	32
13	Erik Hjelmås' compendia in HTML format on GitLab Pages	33
14	Hamburger navigation content	34
15	footer design	36
16	UX of the index page	36
17	Mobile version of a web page	38
18	Problem with older announcements	40
19	Snippet from Mariusz Nowostawski's Gitlab Administration appendix	42
20	Thesis's script for course creation in Gitlab to support private information	43
21	config.sh script snippet	43
22	deploy.sh script snippet	44
23	Source file structure	50
24	Generated nav directory	50
25	Generated nav page markdown file	50
26	Source content folder (left) and resulting website folder (right).	52
27	Command retrieving list of directories	52
28	Multi-project pipeline configuration	55
29	File structure of the GitLab repository.	56
30	File structure of the /webpage directory within the GitLab repository.	56

31	Gitlab CI/CD Snyk configuration	67
32	Snyk vulnerability history	68
33	Snyk issue description	68
34	Snyk report	69
35	Docker official images	69
36	Pa11y web accessibility report	72
37	CodeQuality service report for GitLab	72

1 Introduction

1.1 Task

According to Erik Hjelmås, Associate Professor at NTNU, both the faculty and the student body at NTNU have expressed discontent towards the existing LMS solution in use at the school, BlackBoard Learn. This opinion seems to be shared by the student body and other members of the faculty, according to a survey the group conducted available in Appendix B and interviews with several educators at NTNU, including Deputy Head of Education Department, Frank Alexander Kramer viewable in Appendix C.3. Hjelmås wishes to explore alternative ways of meeting the needs currently handled by the LMS through other technologies already in use at NTNU. He has defined these needs in the form of the following six challenges:

1. Provide easy access to public information.
2. Provide some form of access control to specific files.
3. Publish announcements.
4. Create assignments and receive deliverables.
5. Perform digital tests.
6. Host a discussion forum.

Albeit the existing LMS fulfills all these requirements to a certain extent, Hjelmås wishes to explore whether other systems currently in use at NTNU can solve the same challenges and partially replace BlackBoard. Hjelmås would like the first four of these challenges to be solved through a solution made in GitLab Pages, while the remaining two can be delegated to other technologies and are as such out of scope for this thesis.

The task is to explore the possibilities for recreating certain LMS elements through an alternate service, with a primary focus on using Git and GitLab Pages. The end goal is to discover possible ways to develop the key functionalities outlined by Hjelmås, and whether or not it is even feasible to meet those requirements by using Git technologies at all.

The research questions of the thesis are therefore as follows:

-
- How should a web page with all the necessary open information for an IT course look like?
 - How can information be posted publicly using GitLab Pages?
 - How can a GitLab Pages handle the following forms of access control?
 - Information available to all
 - Information available only to participants of a given course
 - Information available only to the course coordinator
 - How can GitLab Pages be used to deliver announcements to students?
 - How can receiving deliverables and providing teacher feedback be implemented in GitLab?

1.2 Problem area

An LMS is absolutely vital to the modern education system. They represent a platform that allows educators to effectively share information with their students by gathering all the required curriculum, assignments, and grading in a single place. This digitalization of information makes it easier for students to follow their courses and keep up to date with relevant information and deadlines. There are several excellent options for educational institutions to choose from, but most of these options are designed to appeal to as many users as possible. This is not necessarily a bad quality, however there are times where a specific demographic might want a more specialized tool, tailored to their own needs.

BlackBoard Learn - NTNU's current LMS - is subject to complaints from both student body and faculty alike. Some teachers find it to be restrictive, with some settings and functionalities hidden and hard to use. Students experience that it can be difficult to find the content they're looking for in the current LMS structure. For teachers and students in the IT field that are familiar with file-sharing and version control, some of the abstraction and design of the site can be more of a hindrance than a help. This in turn raises a very interesting question: Could any of the existing tools already taught and used at NTNU be used to fulfill the needs traditionally fulfilled by commercial learning management systems?

1.3 Project scope

In order to ensure the group can fulfill the assignment in a good and efficient way, an actionable scope was defined through communication with Hjelmås. The purpose of this section is to clarify what the group must accomplish to answer the assignment to a satisfactory degree.

Scope

- The thesis should explore whether the proposed tasks can be solved using GitLab Pages.
- The developed system should demonstrate the discovered solutions.
- The front-end of the system should have an academical appearance matching that of existing websites affiliated with NTNU.

Limitations

The system is uniquely designed for educators with a base level of IT know-how, and as such there will be a reduced focus on abstracting the teacher's end of the system. As an extension of this, repository developers are expected to have a Linux environment available in which to run certain scripts essential to the system's performance.

Effect goals

These goals outline the desired outcomes of using the finished system, based on the group's conversations with Hjelmås. They serve to guide the direction of the project and will be used to determine both result goals and system requirements.

- Simplify IT educators' teaching experience by offering a tailor-made solution using Git technology as opposed to a generalized system.
- Make non-copyrighted or otherwise confidential teaching materials from a teacher's courses open to the general public.

-
- Reduce costs and increase flexibility by utilizing an open-source solution as opposed to a contract-based product.
 - Minimize risk associated with the website front-end of the developed system.

Result goals

Criteria that should be met for the project to be considered finished are listed as result goals. The result goals serve to help measure the progress of the system throughout development and outlines key points that should be met in order to satisfy the intended effect goals of the project. It is noteworthy that performance is not the focus of the project, moreover there aren't many relevant numerical values to be measured or set as goals. The nature of the thesis is to explore possibilities and develop prototypes, and as such the result goals are more akin to milestones.

- Decide whether or not the defined challenges can be solved using GitLab Pages.
- Develop a prototype that demonstrates a possible implementation of the discovered Git-based solutions.
- Ensure the system is entirely open-source and free to use.
- The project should be finalized by May 20, 2022; i.e. 4 months from project start.

1.4 Demographic

When creating a product, it is important to plan out who the potential customer will be. The target of our task was already defined by our client as members of higher education who have a certain level of IT competency, and who are unhappy with the existing solutions. Hjelmås elaborated that this system would most likely only be used by second year students and higher, ensuring that all users should have a base level of experience with Git technologies.

The demographic is further divided into two main types of users. One is the course coordinator who will be running the GitLab repository. Seeing as their role is to run the repository, these users will be given full Git developer status and are as such expected to have a significant amount of experience with Git and other basic aspects of IT like running scripts locally. The other group of

users is the students. Students will mainly be interacting with the product through the website(s) generated by GitLab Pages, and any interaction with the actual repository will only be required when absolutely necessary. That being said, students are also expected to be familiar with Git.

1.5 Report Structure

This section shortly describes the premise of the different sections of the report.

Introduction

The introduction sets the tone for what our task is about. We describe the task provided, by stating the requirements set by our client, as well as defining limitations of our own in *Project Scope*.

Background

Background serves as the basis for the project, stating information that *is already known*. This section describes the goals set by the group for the project, as well as providing the reader with some insight on the existing LMS solution, before moving on with presenting our own findings.

Methodology

Methodology describes the process from task acquisition until drawing a conclusion. An analysis of the system requirements is also mentioned, where a descriptive version of the provided challenges is defined.

Implementation

Implementation delves into the technicality of the product made for the client. The section covers the system architecture, as well as the internal design and an overview of the implemented features. It presents our researched and developed solutions for the aforementioned challenges, before discussing the security of the new installments.

Conclusion

Lastly, reflection and evaluation of the project is discussed. An assessment of whether or not this is deemed considerable to work with in the future is also brought up.

2 Background

2.1 Field of study

The group is tasked with researching and exploring the ability to use GitLab as a LMS or learning assisting tool. In order to complete this task the group has explored several fields of study. Below is a short presentation of the academical fields that has been the most relevant to the project.

Information Technology encompasses all computer related ventures, and the the combination of technologies and methods used make the project fall squarely under the field of IT.

Educational Technology is learning through technology by a combination of computer hardware, software and educational theory[2]. The project aims to replicate LMS functionalities and is as such considered to heavily entail Educational Technology.

Information Security is part of information risk management and is the practise of protecting information by mitigating risk. The field of information security is always relevant to some degree when developing software, and this project is no exception.

Web Development The end goal of the project is to create a system that generates websites, which necessarily includes the development and design of the resulting site.

2.2 Purpose

The purpose of this thesis is to explore the possibilities of solving certain LMS-related challenges using NTNU-supported tools, mainly in the form of GitLab Pages. The task description only demands the exploration of these possibilities, with no mention of product development being necessary. However, as the project developed further and through further talks with Hjelmås,

it was decided that the group should also develop a functioning prototype demonstrating the solutions discovered.

The main task became to develop an alternative to general LMS solutions, taking into account the requirements posed by the task description as well as facilitating the needs of other IT students who are also dissatisfied with the current platform. This was done in close collaboration with other teachers within the faculty, who have either showed interest in the project or have knowledge and experience relevant to the themes of the task. The group made contact with several educators who had developed their own GitLab Pages solutions, hoping that they would have insights that could help with the thesis.

2.3 Motivation

As an underlying assumption in bachelor thesis, the populace is dissatisfied with current implementations of Learning Management Systems. Interest in researching alternatives for LMS has risen gradually for the duration of being students ourselves. Fundamentally, education has a manifold impact on society. At the individual level it serves as a playing ground to acquire social skills, knowledge and opportunities. On a larger scale education is closely linked to society as a whole considering most if not all of men and women undergo extensive periods of their lives living as students. With this in mind, everyday life of students and teachers are centralized around Learning Management Systems. Because the populace is dissatisfied with the current solutions for LMS, this bachelor thesis serves an important role of presenting opportunities in alternative systems, as well exhibit an exposition of an LMS based on openness of information.

2.4 Project Members

Group members conducting the bachelor thesis are as of writing finalizing their bachelor degree program “Digital Infrastructure and Cybersecurity”. Albeit studying in the same program, each member have a different field of interest and background than their peers.

It is notable that prior academical studies, working experiences and expertise have affected the management and methodology framework of thesis.

2.4.1 Organizing

Group members are organized into the roles of:

- Scrum Master
- Analyst
- Developer

Scrum Master ensures common understanding of values and goals during development, facilitates communication and collaboration between group members which results in steady progress towards reaching project goals. Role of Scrum master was given to Nicholas Bodvin Sellevåg based on experience with managing teams from the Norwegian Armed Forces. In addition, being a developer working under an agile framework.

Analyst accumulates data-driven opportunities to explore during software development. With regards to thesis conducting research is an all encompassing role. Despite this, the primary occupant of analyst was Yan Senko with working experience as a security analyst.

Developer realizes opportunities into a tangible product through coding. Developers are responsible for conducting testing, evaluation of produced material and corresponding documentation. Throughout the project, developing was handled “full-time” by Fabian Kongelf and Oddbjørn S. Borge-Jensen. Nicholas has also provided sufficient support with the solution, namely with the development process of the CI Pipelines of GitLab.

2.4.2 Distribution of Workload

Scrum master conveys the distribution of work by means of tasks in Jira, the work management and issue tracking tool. Tasks are defined at the start of time-boxed periods lasting two weeks named sprints. All group members partake in the exercise and construe a backlog of tasks. Scrum Master presents the main goal of each sprint, Analyst presents opportunities to explore and Developers give insight into the feasibility of such opportunities.

2.5 Experience

The bachelor thesis has been conducted as an experiment to document opportunities for using Git-based systems to solve challenges related to LMS. Exposure of applying scientific research methods to software development became an essential part of the thesis. Furthermore, working experience with methods for mapping requirements, target demographic, dependencies and usage of features to be developed enhanced the thesis as multiple members have or were working as developers or security analysts.

First and foremost, having completed two and a half years of the bachelor degree program “Digital Infrastructure and Cybersecurity” manifested the values, norms and attitude practiced while conducting research into solving the thesis’s research questions. Learning outcomes of aforementioned program is listed as the following: “This course of study will provide you with a much sought-after competence in the field of cyber security, robust data networks and modern IT operations with virtualization and sky solutions; skills that are in high demand in the current job market. The program will provide you with the practical skills within infrastructure and security demanded by the job market. You will learn how to plan and realize virtual assemblies of machinery, interpret the threat situation and become good at monitoring and security” [3].

To conclude, the group members are experienced with developing and operating applications with a scaleable digital infrastructure. In addition, professionally knowledgeable in documenting software development in adherence to scientific method.

2.6 State of the art

2.6.1 Blackboard Learn

Blackboard Learn is the currently used LMS at NTNU as of spring 2022. Developed by Blackboard Inc., it is a primarily web-based solution that can be run either locally or on the cloud in the form of Software as a Service. While not a directly lacking product, it is designed to be used by educators from all kinds of different fields of study and as such does not necessarily align with the needs of an educator from within the field of IT that has significant experience with other file sharing technologies. Most of the heavy lifting is done behind the curtain and allows

very little customization. Likewise, uploading and updating files, as well as the file structure are designed to be easy to understand, as well as practical to use. For IT educators who are used to tools like Git, these can be frustrating roadblocks that serve them no real purpose. Following is a list of strong and weak points of the learning platform from the perspective of an IT educator, and to a lesser extent an IT student.

Strong points

- Overall well-functioning, fulfills all requirements.
- Good hand-in and feedback possibilities.
- Ability to create online tests.

Weak points

- Expensive.
- Enforcement of standard file structure limits teacher's ability to shape their own courses.
- Streamlined - not ideal for teachers with IT know-how.
- UI has too many elements, can be difficult to navigate.
- Content is private by default.

2.7 Related technologies

During the thesis the group used many different technologies of which might not be common to everyone. In this section we will explain these technologies so the reader may gain a greater understanding of the more technical aspect of this report.

GitLab

GitLab is absolutely essential to the project as the task is based on exploring GitLab's features in order to create an LMS. GitLab coordinates repositories (file structures), allowing the user the

ability to create and invite other members to collaborate on projects. Every member can clone the repository and edit its content locally on their own computers. Any local changes a team member makes can be uploaded to the original repository by going through a process called a *merge*, an automated way of combining the changes made in two separate instances of the same repository.

GitLab CI/CD

GitLab CI/CD is a tool for software development. CI/CD stands for continuous integration and continuous delivery or deployment. Through a configuration file (`.gitlab-ci.yml`) GitLab's CI/CD will trigger a pipeline and create jobs of which may create containers and run scripts. the CI/CD can build, test and deploy code in your git-repository [4]. The group use GitLab's CI/CD to create a web-service by create a Ruby container and run the deployment commands for our Jekyll website generator, the resulting web-service is hosted by GitLab Pages.

GitLab Pages

GitLab Pages is one of GitLab's features. Through GitLab's CI/CD pipeline, GitLab pages hosts a webpage under a default Git domain (`*.gitlab.io`), but can be configured to a custom domain [5]. A GitLab page is created by a static site generator or from plain HTML code present in your GitLab repository. GitLab Pages does not support dynamic server-side support such as `.php`. On each iteration (`git add / commit / push`) the CI/CD is triggered and the website will try to deploy again based on the new code added, on success the new website replaces the current site, if the CI/CD fails the current website remains. Figure 1 is a flow chart of the process to create a GitLab Page.

Jekyll

Jekyll is a static site generator, which renders markdown or Textile and Liquid expressions to static web-pages. As Jekyll is a static site generator it does not use databases to dynamically load content, but generate variables containing links, categories, tags and more accessible through Liquid expressions [6].

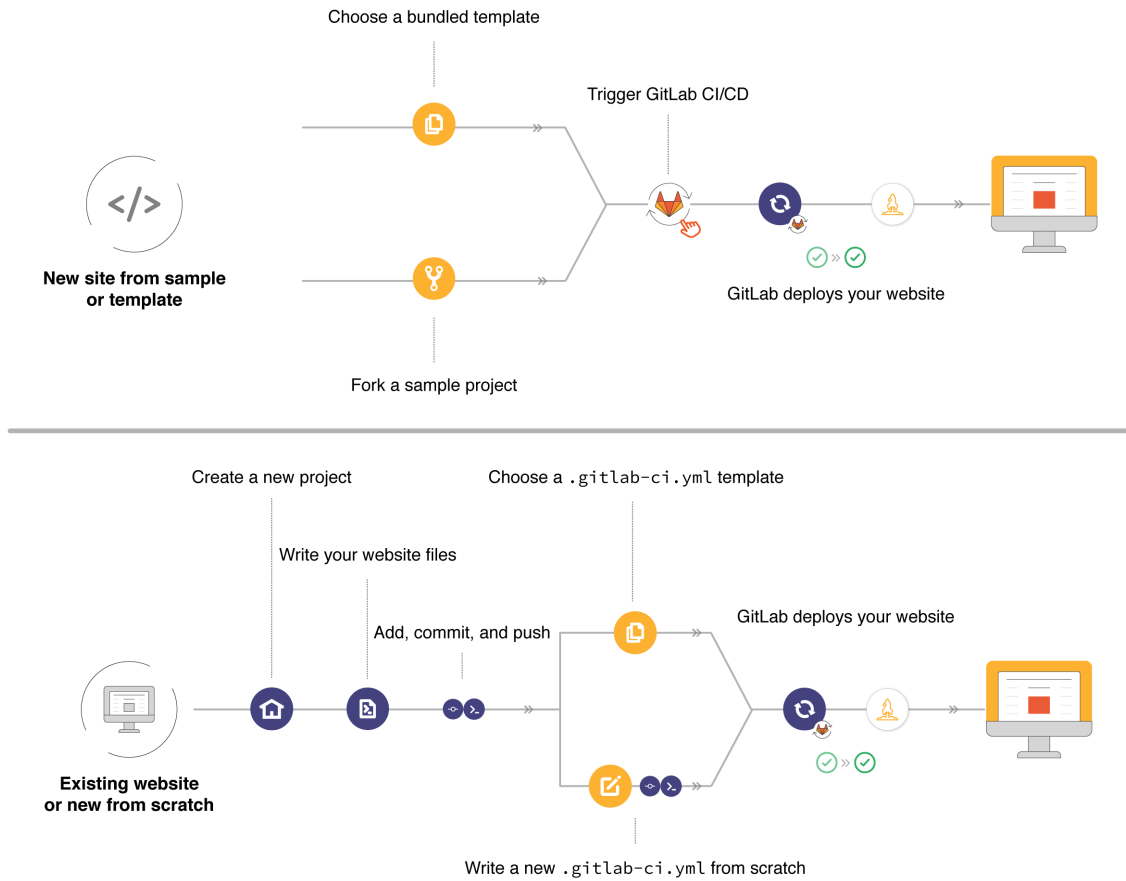


Figure 1: GitLab Pages flowchart

Source: [5]

Markdown

Markdown is a lightweight markup language used to format text using a normal text editor. Markdown formatting uses symbols ahead of the text to define the text snippets format, for instance `”#”` denotes that the following text is a header. When viewing a markdown document an interpreter compiles the document to a formatted easy to read version, similar to how a code is compiled to an executable program.

Code languages

HTML/CSS/JavaScript are languages for website development that allows developers to define the layout of a website, assign each element a visual style, and add functionality and user interactions.

Bash script is a set of instructions to be executed within the Bash shell. There is no difference between running a Bash script or writing the commands directly into the shell itself, the scripts serve only to make the process of repeatedly running the same set of commands more efficient. Less readable than programming languages like C# or Python, but very widely supported.

Ruby is the programming language that Jekyll is written in. No new Ruby code has been written for the project, but it is still essential to the developed system. The Docker instance that builds the website uses a Ruby image in order to run Jekyll, and several Ruby plugins have been used to alter Jekyll's performance to suit the project's needs.

Liquid is an open-source templating language written in Ruby. Jekyll has built-in Liquid compatibility that enables performing certain logic operations within Markdown files before they are converted to HTML. This has mostly been used to include other internal HTML files as elements within the Jekyll-constructed HTML page, preventing redundant code (e.g. including the header and footer in multiple Jekyll layouts) [7].

Abstraction of isolation and containerization

The developed system embodies all research conducted and serves as the answer to thesis's research questions. Moreover, the research questions can be reformulated as "Is it possible to solve the main challenges of an LMS with alternate systems available at NTNU?". The developed system is designed as a functioning LMS on the Git-based system GitLab hosted at NTNU. With this in mind, understanding the rudimentary abstractions that partake in the developed system is a necessity for reading this thesis. For example, the developed system is composed of Docker containers running on a given operating system (OS). To fully comprehend the thesis a background into Processes, Namespaces and Containerization is in order.

Docker is a process contained in its own namespace associated with a set of parameters and limits imposed by Control groups[8]. Processes are a combination of instructions often paired with static data that runs on a processing unit until its termination [9]. Namespaces provide processes with their own system view, effectively what a process can see resides in the namespace type and specification. There are different namespaces in the Linux kernel, which Docker's capabilities are dependent on [10]. The namespaces allow processes to view computer resources related to

networking, file storage, users and more [11]. If namespaces limits what processes are able to see, cgroups limits how much the process can utilize computer resources. Moreover, cgroups allows administrators to specify how much data a process can store in memory, priority to accessing resources for computing and storage, measurements of total resource usage and ability to suspend processes part of control groups. Conjunction of namespaces, control groups and processes are what we call containerization, an abstraction of isolated processes.

There are several benefits of containerization. Containers make use of the host machine's underlying Operating system and bundles applications into lightweight accessible containers. Only the necessary packages and dependencies are part of a container [12]. As a result, containers operate independently of OS and can be deployed to effectively any environment.

Vladimir Baranek is a certified lead author and as of Spring 2022, the current Global Enterprise Transformation Leader at Amazon Web Services (AWS)[13]. With this in mind Baranek is considered as an expert in the field of cloud computing and has written an article that summarizes the benefits of containerization.

“Here is the list of the top nine benefits that help in driving containerization adoption:

- **Reduced cost of infrastructure operations** – There are usually many containers running on a single VM.
- **Solution scalability on the microservice/function level** – No need to scale instances/VMs.
- **Better security** – Full application isolation makes it possible to set each application's major process in separate containers.
- **Instant replication of microservices** via replicas and deployment sets.
- **Flexible routing** between services that are natively supported by containerization platforms.
- **Deploy anywhere** – Including hybrid environments.
- **Full portability** between clouds and on-premises locations.
- **OS independent** – They don't need an OS to run; only the container engine is deployed on a host OS.
- **Fast deployment** with hydration of new containers and termination of old containers with the same environments.

- **Faster “ready to compute”** – Containers are ready to start and stop within seconds in comparison to VMs.

” [14]

The benefits of containers as described by Vladimir Baranek are from the perspective of using virtual machines in the cloud and applicable to thesis as the developed system is exclusively targeted to be hosted in a cloud environment. To summarize, containerization is beneficial in its lightweight medium and security through isolation of processes. Containers contains the dependencies needed for an application and runs on top of host OS, in contrast to fully fledged virtual machines that includes OS configured to run applications. The difference in size between a container and virtual machines is visually presented in Figure 2.

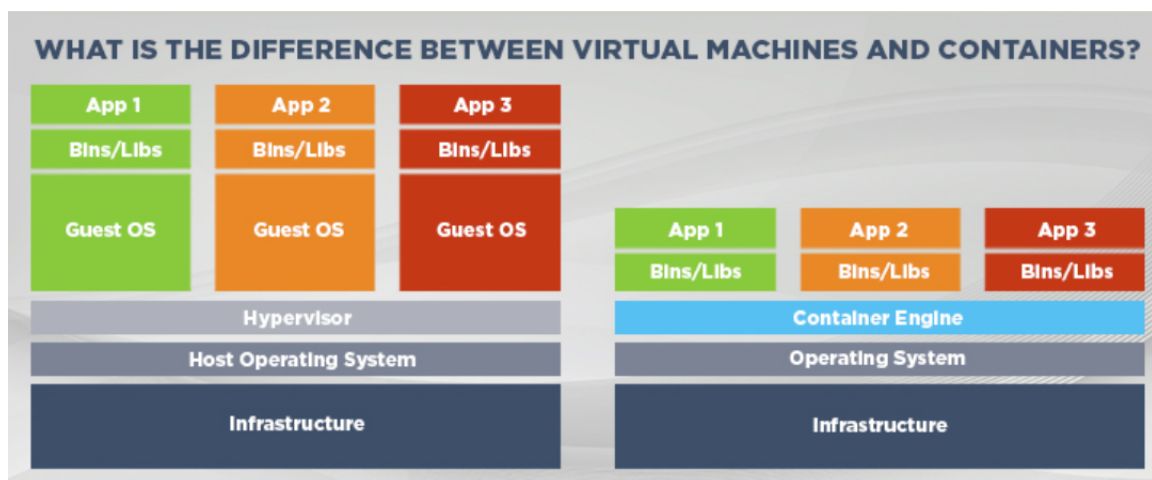


Figure 2: Container VS virtualization

Source: [15]

Comparing linux based containers to virtual machines is visually striking on the additional abstractions layers in place. In fact, removal of abstractions does increase utilization of the underlying computer resources and a considerable benefit of containerization as well as memory usage [16].

3 Methodology

In order to complete the task in a satisfactory way, the group has designed a workflow. The workflow takes inspiration from the scientific method [17], and is divided into several segments,

each with one specific task:

Step one is to find a question in the form of a problem that should be solved. The client has defined the primary challenges of an LMS, each of which is depicted as a problem to be solved throughout the bachelor thesis. Furthermore, defining the challenges and its scope are essential to conducting our experiment and can be found in the related "definition of done" document. The document specifies the problem area, which actors are involved, assumptions and lastly the system features. It is note-while that all functional requirements listed within the system features are to be tested at a later stage.

Step two is to design and perform the experiment. In regard to the bachelor thesis, the development process used to address the defined problems are considered as experiments. For example, initially a development plan is created after the definition of done has been accepted by the client. The plan contains individual tasks that work towards milestones detailed inside the definition of done. Status of development was continuously visible in the issue tracking tool *Jira* that enabled linking issues to code. After the developed solution reached the minimum viable product threshold the product draft would conclude and a demonstration to the client is given.

The last step is to perform tests in order to answer the question. After development, a demonstration of the implementation is held, and the "Acceptance testing" phase begins. The client becomes responsible for acknowledging if the product draft has solved the challenge. As the name suggest, client acceptance of the implemented solution is the end goal.

3.1 System requirement analysis

As described by Erik Hjelmås in the task description the project should fulfill four critical functionalities. These core functionalities are what shaped our system requirements through the entire planning and development stages.

3.1.1 Functionality to create website from markdown

The system should allow a teacher to upload content to their webpage which will be visible to anyone. This means a teacher should have a way to update the content of the repository,

while the public needs a way to view and browse it. As defined in the task description the teacher should be administrating and uploading content to the site by interacting with a GitLab repository, while the public should be viewing the content on an associated website generated through GitLab Pages.

3.1.2 Functionality to support access control

The system should allow a teacher to adjust what content is visible to users. The levels of visibility that should be supported are "teacher only", "all students" and "single student". The teacher should still have access to content marked as "single student" or "all students".

3.1.3 Functionality to publish announcements

The system should allow a teacher to create a special form of content in the form of announcements. These will be the teacher's way of imparting important information and as such they must feature front and center on the site, as well as support some way of alerting students that an announcement has been made.

3.1.4 Functionality to handle assignments

The system should allow a teacher to create tasks for students, and in the same vein allow students to upload files as answers to these tasks. The teacher should be able to close the tasks at a certain time, preventing further uploads or updates from the students. The teacher should also be able to provide some form of feedback for the student's answers.

4 The developed system

4.1 System architecture

A software's architecture is the fundamental structure or platform of which the rest of a software system is built upon. A software architecture is a structure including software elements, their

properties as well as their relations to one another [18]. The architecture represents the design decisions related to the overall systems structure and behavior [19].

The following sections describes the developed system's architecture and entails a high level overview of the information flows and functionalities required in official solutions. Architecture design further outlines the actions and type of actors involved in environment of the system. Actors of a given type are interchangeable, meaning the architecture describes specifically what roles an actor needs to fulfil for the system. As a consequence, the architecture design constitutes how the system should work without regard to how it was implemented. For example, GitLab has been used as a platform for thesis's developed system, but a later revision could try GitHub as the platform instead while adhering to the requirements defined in the architectural documentation.

4.1.1 Architectural requirements

The goal of thesis is to research opportunities in utilizing alternate systems at NTNU for solving challenges typically handled by an LMS service. The architectural requirements are based on the five core functionalities as defined by the client:

Create website from markdown The system should allow for uploading and storing of files that should be viewable by all users on a generated website.

Support access control The system should allow for uploading and storing of files that should only be viewable by certain users.

Publish announcements The system should have a way of informing users of changes or other important information.

Handle assignments The system should allow teachers to create assignments, and allow students to upload files for these assignments. These files should have a traceable relation to both the student who uploaded it and the assignment it is an answer to.

Testing and monitoring Confidence and integrity of software is provided by system with feedback from testing. In addition, system status can be monitored from statistics of test results.

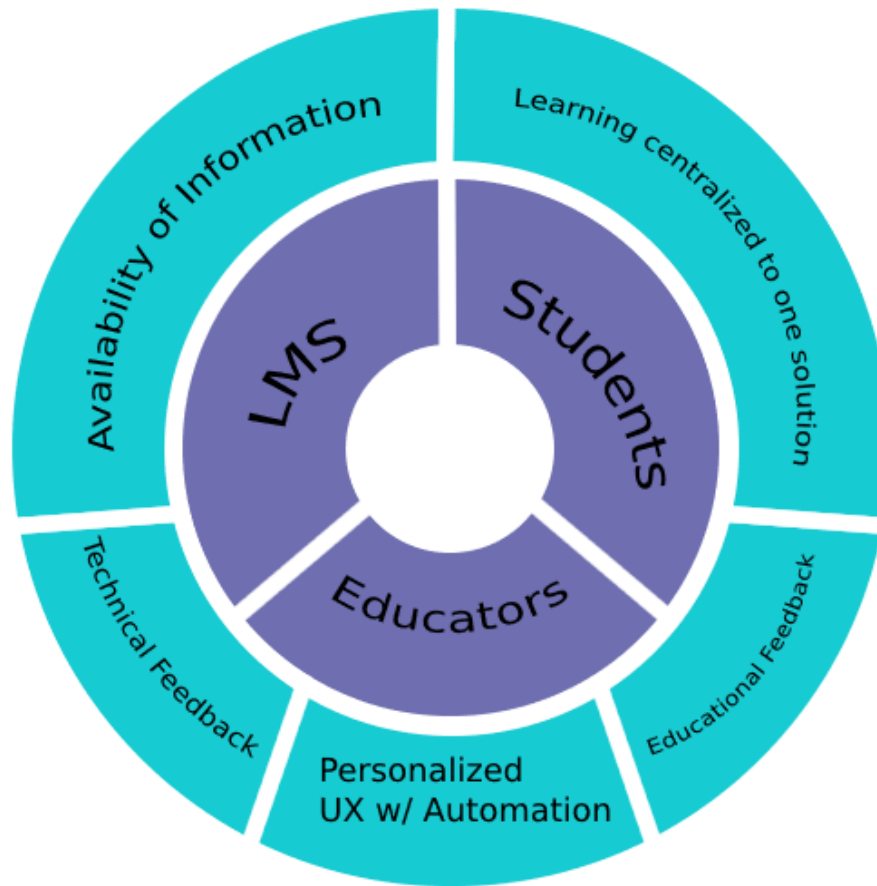


Figure 3: Values of the architecture

The developed system architecture reflects the core values defined by the client and group members. Namely, emphasis on feedback and open information. Software architecture values can be seen in Figure 3. Different iterations of developed system should preserve the stated values in its design to be considered a valid solution. For example, an iteration lacking functionalities to produce feedback on a system level for teachers and user level on students deliverables is considered as inadequate for a commercial or marketable product.

Feedback is part of the system design and processes. For example, teachers can expect the system to provide reports on how to improve the confidentiality and integrity of the system itself. In addition, receiving feedback on how students performed on tasks and assignments is central for determining the overall learning outcome and potential improvement [20]. As a result, feedback is an essential process of developed systems and core value of its architecture.

Further talks with the client brought forward a desire for teaching material to be made publicly available as much as possible, even to those not affiliated with the university offering the course.

Before the rise of LMS learning materials and curriculum used to be self published on servers, readily accessible to anyone. Learning materials was not limited by a students association or partaking communities and the architectural prominence on open information's origin. With this in mind, the architecture for thesis's developed system arose from an IT-teacher's nostalgia towards early internet values, but also the need for a modern, configurable and dynamic system.

The group is of the opinion that modern systems should be designed with scalability and automation in mind. The developed system's architecture is therefore based in the cloud and supports continuous integration and development through GitLab pipelines. One could argue that the architecture is unique in its targeted demographic towards IT competent users and focus on CI/CD. Furthermore, the architecture allows incorporation of the best suited systems to address different challenges. For example, instead of developing a new platform for announcements the architecture enables alternative systems like Microsoft Teams to be used as a communication channel. In essence, cloud based systems with CI/CD can create complex pipelines with interactions between a multitude of systems and solutions.

With reference to existing solutions, BlackBoard Learn has been used as a baseline for core functionalities. Despite this, interviews with students and teachers revealed several key aspects of BlackBoard Learn to be improved upon and incorporated into the designed architecture for thesis's developed system. An assumption written in thesis's task description was that NTNU's installation of Blackboard does not satisfy the professors and students expectations (See section: 1.1). Similar results were found in the sample survey conducted by group members in which several entries from students at NTNU showed dissatisfaction to the graphical design and user experience with Blackboard Learning as well as other LMS systems (see appendix: B). To enable teachers to improve web based course material, the architecture signifies personalized user experiences with automation. In practice, teachers can expect developed systems to be customizable and provide automation for processes that are repetitive with rule based behavior. Moreover, group members believe addressing user experience discontent is achievable through allowing teachers with a background in information technology to make the necessary changes themselves in the editable system. Despite this, an arising problem was enforcing more tasks onto each teacher to update their own installations of the developed system. To address this concern teachers can expect automation of applying styling and handling of documents. Leaving a small subset of tasks for teachers to manually update and configure, specifically the

creation of new or editing existing webpage styling as they see fit themselves. To summarize, the architecture strives to enhance the user experience of teachers administrating LMS systems, in addition improving students web based user experiences as recipients of course material. Teachers provides their own course content as input to the system and it is transformed through automation into personalized websites as output.

The overarching process the architecture enables is input of teaching material and a resulting output of web interfaces. The teaching material in itself is therefore a vital and needed component of the system. The content provided by teachers will vary in outline, purpose and information. For example, the content can be categorized as tasks, solutions, announcements and subject matter. Furthermore the information might contain copyrighted material which is not appropriate for being publicized publicly. As a consequence, a segregation of information is needed whereas most information is publicly available while private information is still made available to student or authorised personnel. To create the segregation of information the system requires a list of students and teachers to enforce appropriate access control. Similarly, all challenge areas has its own set of dependencies and assumptions for working properly. The following subsections elaborates on the architectural design in each challenge area.

Functionality to create website from markdown

Public information should be provided through openly accessible Web interfaces on the internet through a server-client relationship. Moreover, students must as a minimum be enabled to requests a server's web interfaces to view course content on any device with network browsing capabilities. All public information in relation to a course is accessible on a centralized web interface. Preferably design and styling of web interfaces shall reflect the areas of interest from an IT students perspective. Navigational elements on web interfaces ought to guide students intuitively through a given course.

Functionality to support access control

Private information builds further upon the base built to support the public information. In order for the system to successfully support private information there must exist a way to for the teacher to make certain data available to only certain users. All files should always be visible to

the teacher.

Functionality to publish announcements

Announcements consist of three major functionalities that have to be supported by the architecture. The starting point is to have somewhere to store the actual announcements in a way that differentiates them from other content. There must also be a way for teachers to create and upload the announcements themselves. Third, the system must have some sort of channel through which it can reach its users, either through something direct like e-mail or third-party software like Microsoft Teams. An addendum to this third point is that the system must also be able to detect when a new announcement has been made in order to communicate it to the user base.

Functionality to handle assignments

The main necessary components required to support deliverables as they pertain to this project is a server and a client through which users can interact with said server. The server is responsible for keeping track of the assignments and each individual students' answer to each assignment. Ideally the server should contain some kind of solution close in function to that of a relational database, allowing each answer to be related to both the student who delivered it and the assignment it is an answer to. As Figure 4, the client serves as the front-end the user interacts with and should allow the user to both log in to the system and upload files as answers to specific assignments.

Confidence and integrity

Teachers should gain confidence in the developed system through automation. Course administrators utilizing the developed system shall receive automated reports depicting the overall status of system's health, security and accessibility. In addition, automation in general is required to be scalable as if each automation is a component of a larger system. As a result, continuous integration and development is achieved on a architectural level.

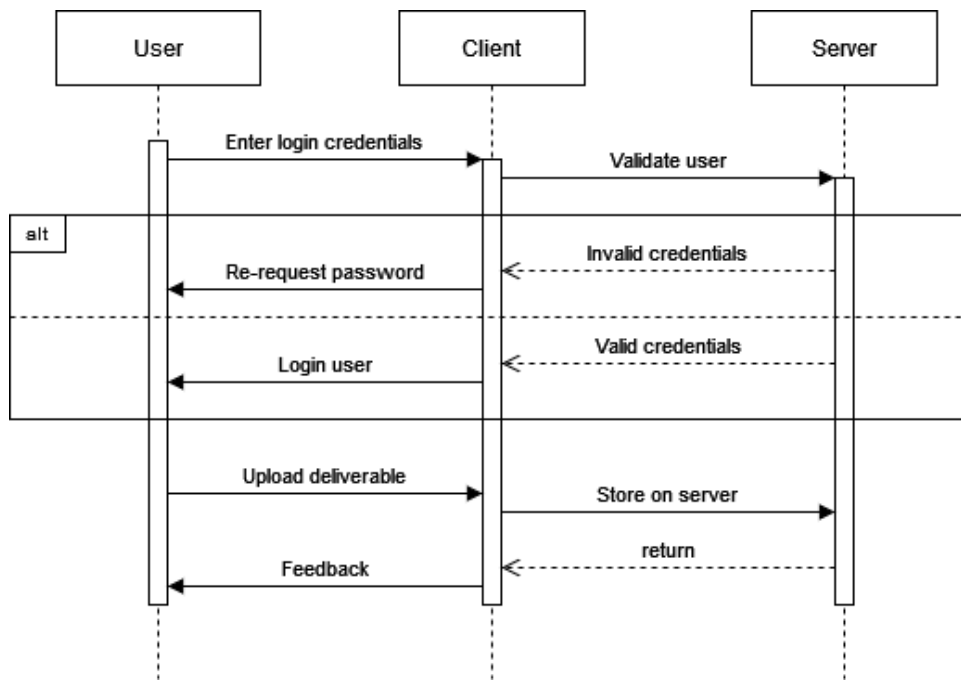


Figure 4: Sequence diagrams for deliverables

4.2 System Design

This section is set to describe the different processes within the GitLab solution, delving into the steps taken from start to finish, with a technical standpoint.

- Availability of public information
- Exclusivity of private information
- Engaging announcements
- Deliverables
- confidence and integrity in software

4.2.1 Information Publication via GitLab Pages

The GitLab Pages workflow is designed to automate as much of the work as possible, requiring minimal effort from the publisher. As to commemorate the task given by the client, the publisher in this case will be a *teacher*. In the case of a teacher wanting to publish information to the public, the workflow will only require them to upload the desired documents from their local repository

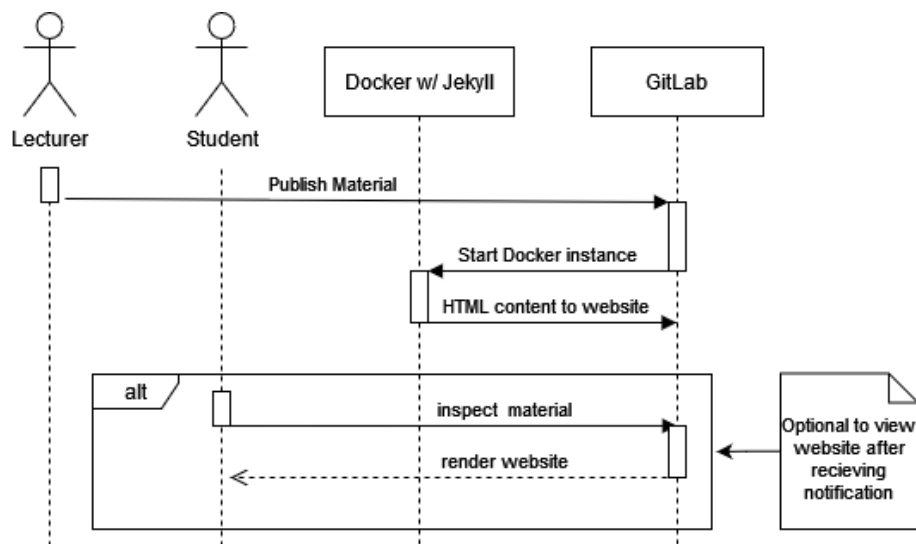


Figure 5: Sequence Diagram of website generation

to the cloud, using the GitLab CLI commands *git add*, *git commit* and *git push*, respectively. This is the last manual action the teacher does, before the automated course of actions takes place.

After the repository update goes through, the automatized part of the process begins. It is initiated by a process runner, which reads a YML-file specifying the CI process, and performs the listed instructions with included scripts accordingly. Figure 5 portrays a sequence diagram of the publication workflow.

When it comes to the deployment of GitLab Pages, the “Deploy” pipeline is designated for just that. Upon startup, the designated Docker Container firstly installs all required dependencies, before running a shellscript which takes care of the build process. One of the dependencies is Jekyll, the service responsible for the webpage design. The script starts off by replicates the newly created content into the webpage content directory, for conversion and later publication; However, before Jekyll can publish any content, it needs to *detect* it. The configuration file *_config.yml* determines the content folder in which Jekyll is required to search through. It is obligated by the lecturer to publish their material there, or else the service will not detect it. As the pipeline starts, the Jekyll-initiated scanner will look through the published files, and convert every Markdown file into an HTML page. Files of other extensions are unchanged. Finally, when all Markdown documents have received their HTML counterparts respectively, the HTML pages are then displayed on the Docker-ran GitLab Pages. Articles follow the same file hierarchy as the one specified in the */content* folder of the repository. From this point on,

the students can access the material. The container does not host the webpage, however; This is a task done by the GitLab instance, which takes care of the hosting. By infusing the pipeline job metadata with the tag “pages”, GitLab automatically interprets the process to be regarding GitLab Pages.

4.2.2 Exclusivity of private information with roles

Exclusivity is achieved with role based access control. With respect to Mariusz Nowostawski, Associate Professor at NTNU, group members have created an automated hierarchical project structure that mimics information exclusivity obtained with Mariusz Nowostawski’s own GitLab course implementation as a reference (available in Appendix C.1). Moreover, the developed system utilizes different configuration of projects in GitLab where users have different roles based on which level the information is portrayed in within the project hierarchy. For example, Figure 6 represents the automated project groupings created with the developed system:

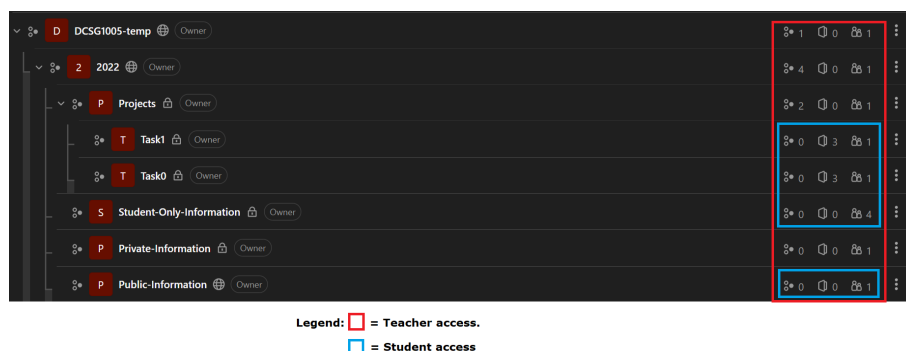


Figure 6: GitLab project hierarchy access levels

Students are added into specific projects they need access to. As a baseline, all GitLab users can view information in public projects marked as a globe. In contrast, only users part of a private project can view the private project’s content. The actions available for users in any given project is in accordance with their GitLab role. To view the full extent of access levels configurable in GitLab users are advised to use GitLab’s own documentation [21]. In practice, creation of the access controlled project hierarchy is done as follows in Figure 7.

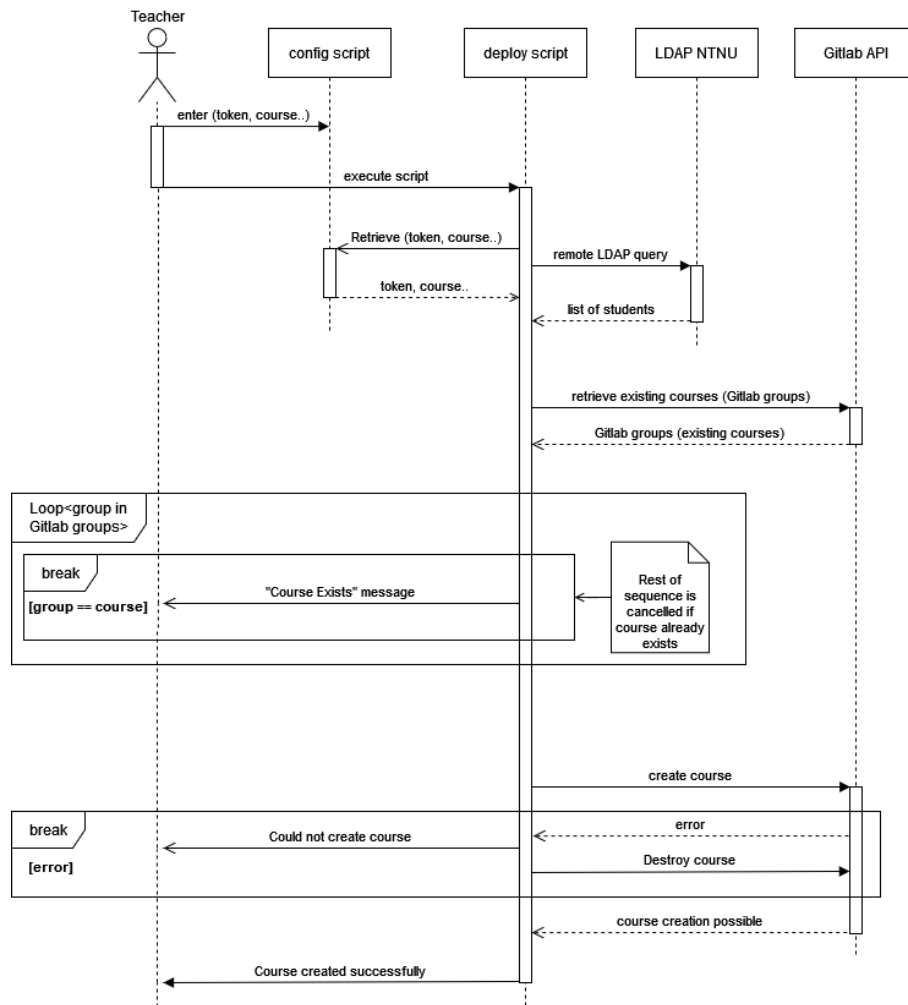


Figure 7: GitLab project hierarchy creation

4.2.3 Engaging announcements through automation

After the teacher has published their content on to GitLab, an alternative process will start, designed to notify about the changes on a Microsoft Teams channel. The "Notification" pipeline of Figure 8 is responsible for publishing the message from teacher's aforementioned *git commit* to a designated Teams channel, with the use of a connected webhook.

The pipeline firstly reads from the Git CI/CD generated variables, in search for metadata regarding the commit. The pipeline script collects the written commit message, timestamp of published message, the web URL for the webhook service and if the committed file is an announcement, the files content is added to the collection. With the collected data, the script is able to send a webhook request to Teams, which if the connection is established, will successfully publish an announcement on a designated Teams channel made for said webhooks, and accessible for students to inspect. The sequence diagram displayed on Figure 9 portrays the steps taken between

the actors when a publication has occurred.

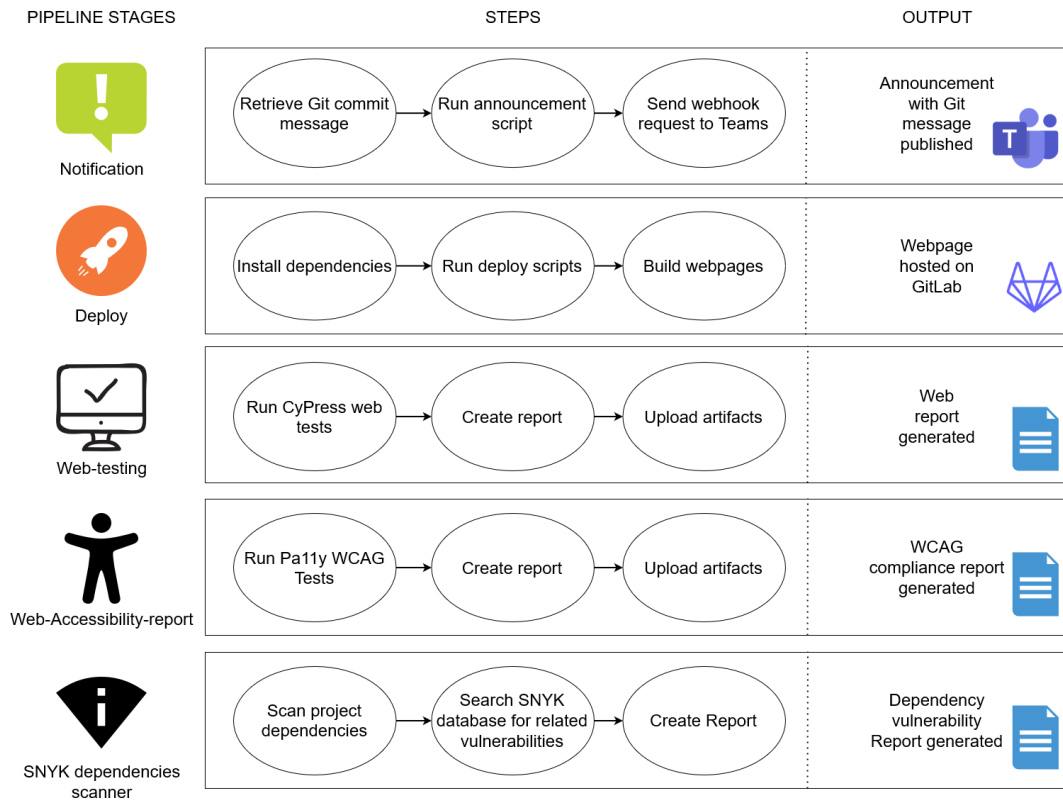


Figure 8: Flowchart of five independent Docker Containers

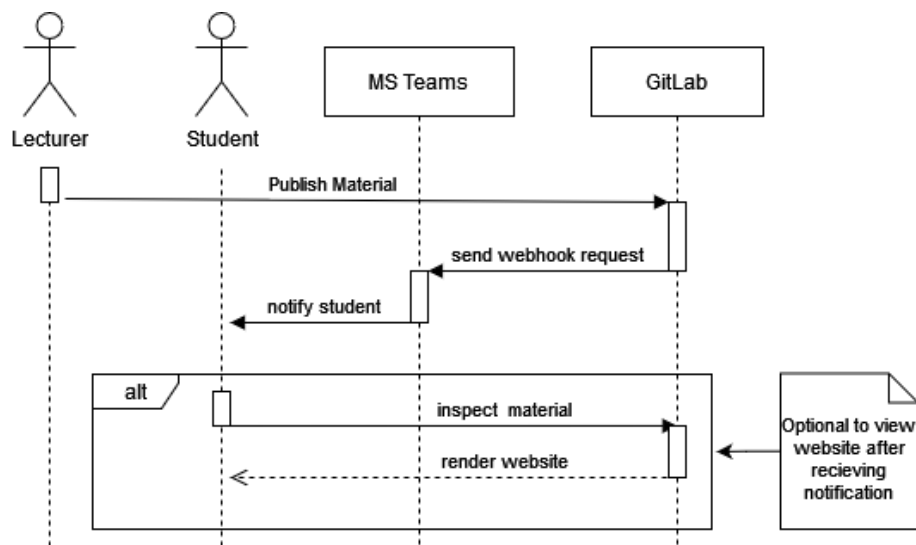


Figure 9: Sequence Diagram of the "Notification" pipeline

When the announcement script sends a webhook request to Microsoft Teams, the request is in form of a curl message to the webhook url. Figure 10 is from the announcement script, it shows the curl message sent to a webhook. The script sends a message based on variables defined earlier in the script, the variables are dynamically allocated based on the commit a teacher makes

when uploading a file. Words starting with \$ are variables, the variable named webhook acquires it's value from the _config.yml file, the other variable is collected form GitLab's CI/CD and consists of metadata associated with the commit.

```
output=$(curl --location --request POST $webhook \  
--header 'Content-Type: application/json' \  
--data-raw '{  
  "@type": "MessageCard",  
  "@context": "http://schema.org/extensions",  
  "themeColor": "0076D7",  
  "summary": "'"$message"'",  
  "sections": [{  
    "activityTitle": "'"$message"'",  
    "activitySubtitle": "'"$time_stamp"'",  
    "activityImage": "https://about.gitlab.com/images/press/",  
    "type": "TextBlock",  
    "text": "'"$announce_content"'",  
    "markdown": true  
  }, {  
    "facts": ["'$facts'"],  
    "markdown": true  
  }, {  
    "facts": [{  
      "name": "URL",  
      "value": "'"$web_url"'"  
    }]  
  }]  
}' )
```

Figure 10: Code snippet from sendAnnouncement script depicting the curl request

The curl request (Figure 10) gives the message card seen in Figure 11 as a result on Microsoft Teams. The message's header is the commit message, below the header is a date stamp of the commits publication. Below the initial header section is a list of all the files created, edited or deleted and the amount of changes made to the file. The number at the right of the file name and path states the amount of change. To the right of the number is a series of plus (+) and/or minuses (-), the plus indicate a added line while the minus is a removed line, when the number is large the pluses and minuses are at a ratio rather than an indicator per line.

4.2.4 Deliverables

The current solution does not feature deliverables, but a system where students can deliver work and receive grading from the teacher is possible and a version of it would work for the users in

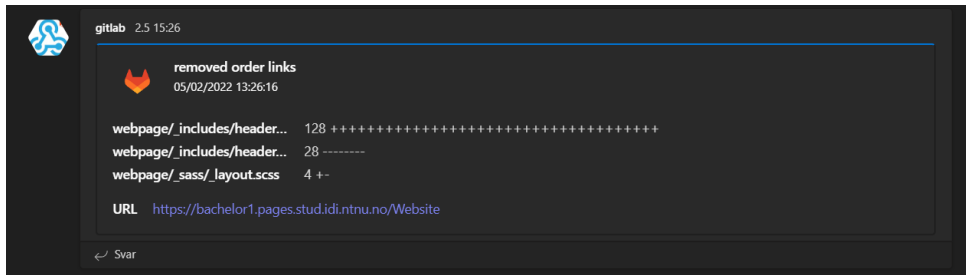


Figure 11: How the announcement looks on teams

the following way:

A student have access to a structure of Git groups, of which a group consists of projects given to students. The student can navigate to this group and find their project, within which they can do whatever they want: produce whatever desirable content and hopefully complete tasks given by the teacher. Students should be able to see task on the website and be able to read the task description.

The course teacher will be able to define the amount of deliverables by changing the GitLab structure. For each task, a group of multiple repositories is created, with each student having their own repository. When the delivery deadline arrives, a teacher can activate a script to create a folder of all the GitLab projects the students have created on their own computer, representing the deliverables.

4.2.5 Confidence and integrity of software through testing

The solution is not only designed to host an educative infrastructure, but also to look within and determine whether some aspects of the design can be improved. The last three pipelines of Figure 8 take this exact responsibility: to test out different aspects of the GitLab repository and generate feedback on what can be done better. They are run simultaneously with “Notification” and “Deploy” pipelines, each within their own Docker container.

The first testing pipeline, labeled “Web-testing”, uses Cypress, a web testing framework, to evaluate the website’s code quality. Our self-made test ensures that no links on our GitLab pages are broken, and that they remain fully-functional upon every commit. The end result, as the figure portrays, is a report generated at a designated folder within the repository, concluding whether the test was successful or if any errors occurred in the meantime.

The second to last pipeline is responsible for ensuring that the website is compliant with the WCAG standard. Pa11y's CI WCAG test involves an accessibility test runner which reviews a provided URL, and generates a report, stating segments which do not correspond with the standard [22]. As for the pipeline in this project, the provided URL is to the GitLab pages. The accessibility tool references the official guidelines for when commenting on errors, with links to the official website for further reading. Another benefit is its configuration options, presenting it as a flexible tool for website accessibility testing.

Finally, the repository will run a SNYK vulnerability scan, focusing on the used dependencies within the project. This open-source installation inspects the dependencies used by the project and investigates whether the installed versions have documented vulnerabilities, as documented on their own frequently-updated database of vulnerabilities and weaknesses [23]. The end report displays the findings of the test, as well as criticality and mitigation recommendations on found vulnerabilities on the system.

4.3 Page design

One of the initial research questions of this thesis poses the question of what a website with all the necessary information to hold an IT course should look like, from a student perspective.

For this purpose, the solution presented is a framework that automatically generates websites with rule-sets based on feedback received from student surveys. Moreover, the developed system enables teachers to freely publish Markdown documents which are processed into an according HTML version made available as websites for students. During the process of creating websites from Markdown documents three significant stages are performed:

- Interpretation of Liquid template language expressions in files.
- Converting Markdown to corresponding HTML and Sass into CSS.
- Populating Layouts with converted and interpreted content.

In relation to the research question, presentation of page design is realized through CSS in the second stage when documents become rendered. Specifically, CSS depicts which colors to apply, fonts, sizing and to some extent placement of Web elements. CSS further enhances the

developed system with adaptive presentation based on the type of device used to browse published websites. For example, smaller screens often needs a different presentation in contrast to larger devices. All in all, CSS remains the means of visually presenting pages and its content. Despite this, the developed system is created with scalability and automation in mind. Manufacturing and mainting an all encompassing CSS solution for websites can be difficult and troublesome with sizable amount of pages. Instead, CSS is used to generally define formatting of HTML while Layouts create different views of any given website in a manageable fashion.

The home page is created from a Layout and displays several pieces of information, including announcements, the about section, a complete table of contents, and the course coordinator's contact information.

The research question also emphasizes designing websites from an IT-student's perspective; that is to say a student who's familiar with web based technologies like Git and their capabilities. To disclose and acknowledge aforesaid students opinions the rules that determine layout and design of websites was reached from the conducted student survey (needs sourcing for attached survey) and group members expertise. The resulting configurations related to styling of websites can be found within “_sass” and “css” folder.

To conclude, the system is configured with a minimal set of assumptions to prevent limiting the content published by teachers. For example, dynamically producing uniformly designed websites was solved with layouts and Syntactically Awesome Style Sheets. Ensuing sections elaborates the implementation and usage of page design elements.

4.3.1 Visual Design

With the student perspective in mind for the User Interface (UI), the design is derived from the notion of minimalism, as well as taking inspiration from NTNU's official website layout. The key getaway has been to limit the amount of elements a student can interact with, to hence improve readability and reduce unnecessary components.

The goal is to create a UI as user-friendly as possible. As the site consists of markdown converted HTML files of which a teacher/professor have created, we have minimal control of the UI, though it compensates with giving us complete freedom on the layout. As we have no control of a

page content, our design accentuates interaction design and visual design. to enforce interaction design we made a rule of thumb, where we focus on achieving tasks in as few clicks as possible, as for example to navigate between any two pages. However, visual design is also important and the page should not be cluttered with links and semi-relevant bits of information.

The website utilizes a static site generator called Jekyll to convert Markdown files to HTML. When imported to the project, Jekyll came with a standard layout, which we used as a base for our own design.

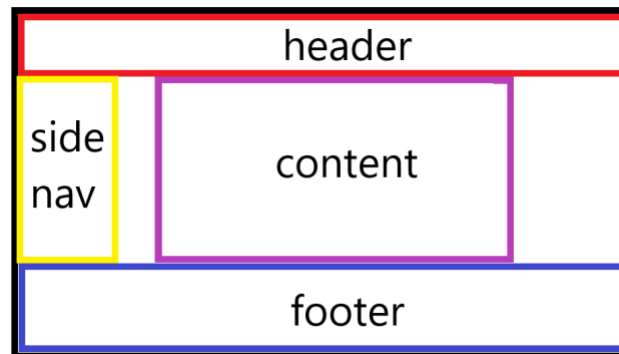


Figure 12: The LMS's semantic element design

On Figure 12, the layout of the website is presented. It consists of a header, sidebar, footer and a content section. The solution has implemented a side navigation bar, header and footer while the content section consists of the markdown converted HTML pages, with the sole exception of the index page.

Our project is an LMS for a university, an important aspect is then the university's visual language. To keep in line with NTNU's visual language, color of links and background colors are similar to NTNU's website colors, Figure 13 displays the current visual language of the website.

Header

The header located at the top of the webpage. It is often considered "sticky", meaning it is locked at the top of the screen and not moving when a user scrolls. The project uses a sticky header-element containing a hamburger-navbar button, name of the course, and a picture of NTNU's logo. The name of the course doubles as a link to the index page.

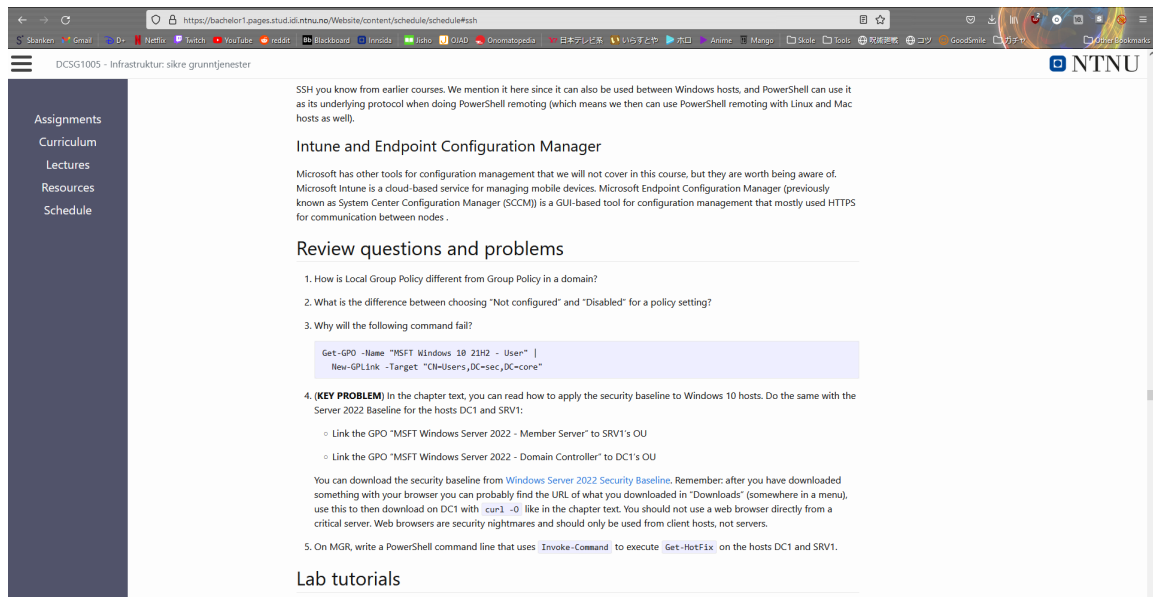


Figure 13: Erik Hjelmås' compendia in HTML format on GitLab Pages

Hamburger Navbar

As a way to minimize the amount of clicks between two pages while maintain a minimalistic visual design, we developed a hamburger navigation bar/menu. The navigation bar consists of a button with three bars stacked on top of each other like a hamburger, hence the name. When the hamburger button is pushed, a menu covers the entire page, and contains the same link structure as seen on the index page. It allows you to navigate to every page on the site. This type of navigation bar also scales well with smaller devices such as phones and tablets, as the entry point is just a button and the menu covers the entire screen. Figure 14 displays the websites open hamburger navigation.

Navigation sidebar

One aspect of the existing BlackBoard design that was deemed good as-is and with no need of changing, is the prominent navigation menu on the left side of the screen. Taking inspiration from this, we made our own, very similar sidebar. It is made to be highly visible and always present on screen no matter where in the site hierarchy the user is. The links present in the menu represent the directories directly contained within the `__content` folder. This means it supports use of both the default folder structure we provide with the product as well as any self-defined structure a teacher should want to use to best fit their courses.

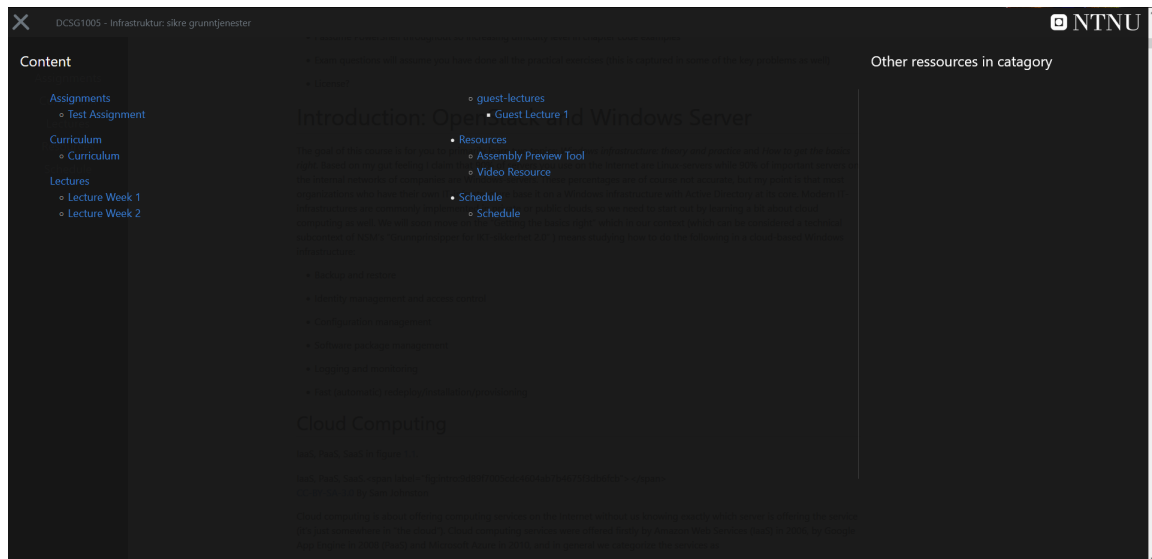


Figure 14: Hamburger navigation content

The default structure was designed based on the group's own experiences, a survey of other students, and existing solutions like BlackBoard Learn and Frode Haug's course sites [24]. This resulted in the following default links:

Assignments contains all obligatory tasks given throughout the course, as well as any other work that the course coordinator deems necessary.

Curriculum contains the curriculum for the course.

Lectures contains recordings of held lectures should they be recorded, as well as viewable versions of any presentations used.

Resources contains additional resources relevant to the course that are not directly part of the curriculum. This can be anything from additional reading materials to external websites or videos.

Schedule contains the course schedule, detailing upcoming lectures, deliverables, and other relevant information.

The same script that creates the sidebar also generates a series of navigation pages that are used to navigate the website. These pages are generated for every sub-directory of the content folder, making up a complete navigation web for the site without using any dynamic components. The navigation pages are very simplistic in design, containing only the title of the folder and links

to each of its contents - both files and further sub-directories. Directory links have a preceding folder icon to help the user differentiate between files and folders.

Footer

A footer is the last element you see when you scroll to the bottom of a web page. Web sites footers typically contain [25]:

- Authorship information
- Copyright information
- Contact information
- Sitemap
- Back to top links
- Related documents

Variable connections

The footer consists of variables defined in the website's `_config.yml` file. Through Liquid the configurations variables are displayed in the footer.

```
<li>Emneansvarlig <a href="mailto:{{ site.email }}">{{ site.email }}</a></li>
```

Above is a code snippet from `webpage/_includes/footer.html`. It shows how a “Emneansvarlig”, subject responsible, usually the main professor in a subject, email is included in the footer. Code encapsulated in double curly braces, `{{`, are liquid code. Liquid command `site.email` finds the site wide variable email, which in this projects case is located in `_config.yml`. In Figure 15 you can see the footer design.

Index Page

An index page, or often referred to as a home page, is usually the first page you see if you enter a website (some sites use a launch page). In this project the index page is similar to an LMS's



Figure 15: footer design

index page and contain announcements, an *about* section and links to the sites content. In Figure 16 is the current design and layout of this projects index page.

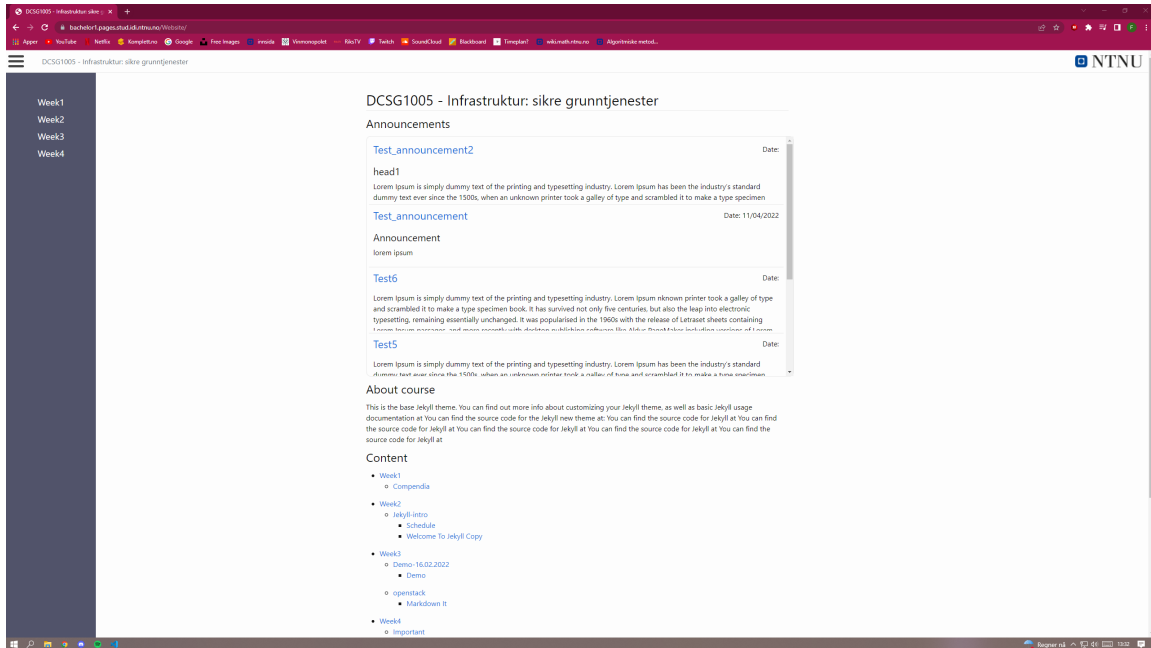


Figure 16: UX of the index page

Announcements

If a professor pushes a markdown file in the announcements folder (content/announcements, the folder name can be changed if also changed in `_config.yml`) the file is viewed as an announcement. On the index page the announcements are listed out as a link to the file and the first three lines of the announcement. A user can then click the link to be moved to the announcement file and view the whole announcement.

About

The about section is paragraph imported from the `about.md`, the purpose is for a professor to write a short exposition on the subjects, the subjects learning material and how the subject will progress. This section should give insight to a user on the websites content.

Index-links

From the index page a user is able to reach every other page on the website, every page should therefor have a link on the index page. Every file is given a link with a name equal to its filename, the links are created in a structure of links which replicate the folder structure where the files are located. A professor can decide how the linking system looks by changing the names of folders and files. This linking structure is then imported to the index page and hamburger navigation bar.

Mobile

One participant from the survey stated:

“[BlackBoard Learn] appen funker ganske dårlig. Systemet generelt er ganske rotete. Er ikke så viktig med uendelige funksjonaliteter, men mer hvor ryddig det er på nettsiden og hvor lett det er å finne fram til informasjon.”

(The [BlackBoard Learn] app doesn't work very well. The system is generally messy. Having as many functionalities as possible isn't what's important, but rather how clean and easily accessible the information is.)

Brian Rashid, marketing expert, wrote for Forbes and reported on an increase in mobile usage and the benefits of responsive web design. The survey confirms his statement where a lot of the feedback was directed to poor mobile integration. The website is design to be responsive, this means the site will adapt to all screen dimensions, which allows users of different media to use the LMS. In the future if phone and tablet screen size become larger, the website is design to be responsive and functional to changes in screen dimensions. The importance of responsive web design can be seen in the quote from the paragraph above, A responsive site will not add clutter, and help create a cleaner visual design, and a seamless user experience [26].

```
// This class defines the content elements width
.page-content {
  padding: $spacing-unit 0;
  margin-left: auto;
  margin-right: auto;
}
```

```
@media screen and (max-width: 1000px) { .page-content { width: 95vw; } }  
@media screen and (min-width: 1000px) { .page-content { width: 1000px; } }
```

Above is a code snippet from `website/_sass/_layout.scss`, it shows the general scss rules given to the content element. Contents width is responsive by adapting to various screen sizes through media queries, if the screen width is larger than 1000 pixels (px), content has a width of 1000px, if the screen is smaller than a 1000px, content has a width of 95 viewport width (vw) or 95% of the users screen width including potential scroll-bars (if a width value of 95% is given, the width will change depending on whether the page has a scrollbar or not). Viewport width is the width of a web browser, while a % value based on parent elements, thus can be larger or smaller than the browser's viewport, when given to the body or the root/parent element the value is based on the browser space. A width of 95vw will span almost all of the screen, given that the content element usually contains information the user is after, spanning the entire screen dimension makes it easier for the user to retrieve the information. On Figure 17 you can see how the site looks on a phone.

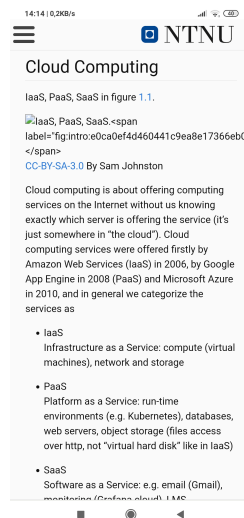


Figure 17: Mobile version of a web page

Variables

The scss code is variable based, meaning the `main.scss` only declares variables like different colors, font and some spacing. The variables are interchangeable, giving an administrator the ability to easily change the color schema of the website.

Below is a snippet from `css/main.scss` where the variables are declared and a snippet from `website/_sass/_base.scss` where the variable is used. In `main` the brand color is set to blue. This color is then given as a default color to all links on the site.

```
// Some of the variables from css/main.scss
$text-color:          #111;
$background-color:    #fdfdfd;
$brand-color:         #2373db;
$black:               #30313f;

// Link style, on _sass/_base.scss
a {
  color: $brand-color;
  text-decoration: none;
  transition: .3s;

  &:hover {
    color: darken($brand-color, 40%);
  }
}
```

How the design suffers from data persistence

The project is designed to be forked, deleted, generated and regenerated easily. Data persistence becomes an issue, how does the developed system keep data from the different instances of the project? for instance, if a file is added to the repository and the website is regenerated, a script can add a date variable to the newly added file, however this edit is within a docker environment thus not editing the original file. If then another file is added and the website regenerated the date variable is replaced with the original document and the data lost.

On Figure 18 is the announcement section on the index page. Announcements are important to date and sort correctly, otherwise an student/end user will have a difficult time finding the newest and most pressing announcement. If a student cannot find an important announcement, they

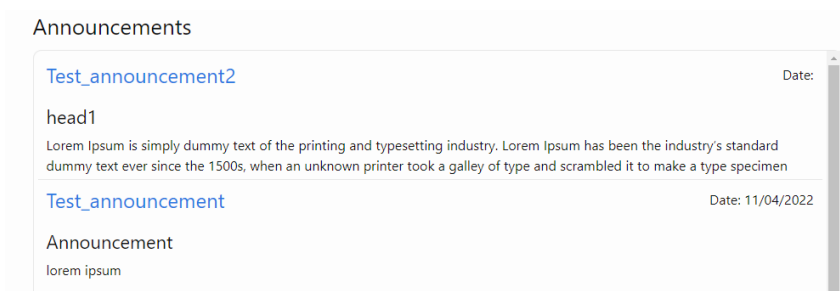


Figure 18: Problem with older announcements

may miss out on changes to deliveries, exams, lectures and so on, which can have catastrophic consequences for the student.

4.4 Implementation

4.4.1 Functionality to create website from markdown

In order to determine what files should and should not be used for building the website, Jekyll demands a strict file structure that requires directories with very specific names. While it's not too complicated, it can nonetheless be a user unfriendly element that affects the overall user experience. For this reason a few additions have been made to the basic structure in order to abstract this element of the system.

At the top level of the repository there is a directory simply named *content*. The system has been designed entirely around the idea that an end user should never have to interact with any other directories than this one, though they are of course still free to do so should they wish. The *content* directory is where the source markdown files and related assets should be uploaded. Any sub-directories created within the directory will automatically be converted to categories within Jekyll as the site is being built. These categories are what make up the table of contents on the index page of the final website.

In order for Jekyll to find the source markdown files for the site they have to be located within a folder with the name *_posts*. The developed system have abstracted this requirement entirely by automatically altering the file structure to abide by the standards Jekyll expects upon site generation. The actual generation happens on a Docker instance and as such the changes made are not reflected in the original repository. The intent behind this abstraction is to make the user

experience more straight-forward and allow for full use of the service without requiring any prior knowledge of how Jekyll works.

The developed system also support users who want to designate their own `_posts` directories and take full control of the Jekyll structure. The automatic system always checks for whether the `content` directory is already Jekyll compatible, in which case it will leave the file structure as it is, making no changes. This way users can choose to either go with the easy, abstracted method or rather choose to deal with the technicalities themselves, without having to change any manual settings or preferences.

For several of the additional functionalities that weren't part of the standard Jekyll process it was necessary to develop new solutions through new code. There were multiple options to choose from when it came to programming languages to use. Shellscrip was chosen due to it's wide compatibility and the fact that the group had prior experience with programming in shell. Furthermore most of the tasks revolved around performing operations on and searching through text, something shell is quite apt at doing through commands like `grep` and `awk`.

There were two other language options that we ended up moving away from. The first of these was Ruby, which is the language Jekyll is written in. The main reason group members chose to avoid using the language was lack of experience using it, and posed a potential risk in utilization of time as we could already solve all our problems in shell instead. The other alternative was another language used by Jekyll, namely Liquid. Liquid stands out from the other alternatives as it is not a complete programming language, but rather a template language. Its utility was exemplified through enabling group members to access the internal structure of Jekyll - such as categories and tags - as well as perform programming logic inside markdown files. While this sounded promising initially group members found that the syntax quickly became convoluted when handling retrieving information from complex data structures, and it also rendered group members unable to refer directly to the file structure in the Git repository, which was important for automation in general.

4.4.2 Functionality to support access control

Exclusivity of information on a user based level was achieved with reference to Mariusz Nowostawski's, Associate Professor at NTNU, take on role based Gitlab administration. The group

conducted interviews with Nowostawski based on his experiences with several courses at NTNU that applies his user access control paradigm (see Appendix C.1). The group's own implementation of the proposed project structure with approval of Mariusz Nowostawski can be seen on Figure 6.

Groups and sub-groups

Groups and subgroups allow hierarchical projects organisation such that logical groupings for maintenance and long-term archiving can be achieved. Below is a proposed scheme:

administration

Group for system administrators. Projects visible ONLY to system administrators.

course

- course - top-level group for all course-related projects
 - imt3673 - top-level group for all imt3673 related projects
 - 2018 - subgroup for archiving purposes for projects from that particular year. This forms a namespace for the concrete course projects
- course/imt3673/ The structure under the namespace, is up to a given course responsible. The structure might look as follows:
 - 2018/imt3673-lectures - Wiki, issue tracker for the lectures
 - 2018/imt3673-assignment1 - repo and sub-project for the course IMT3673 for assignment 1
 - 2018/imt3673-admin - for teachers only, administration tasks, etc.

Figure 19: Snippet from Mariusz Nowostawski's Gitlab Administration appendix

Most importantly, the project grouping is hierarchical. This allows for different levels that users can inherit their access privileges from. For example, at the bottom level individual users within a project are able to edit or view documents based on their role. A full listing of all actions a user has permissions to can be viewed in Gitlab's own documentation [21]. Furthermore, student deliveries are considered as private information and each student in a given course would consequently have their own private project. In this example, the student would have full access to edit and view their own documents. Some courses include peer-reviewing between students and could be achieved with having two students become part of each others projects but only with the role to view their opposing student's documents. In essence, Gitlab administration could be achieved from following the guidelines set by Nowostawski's proposal seen in 19. Despite this, the process of administrating a course through Nowostawski's setup is as of today a highly repetitive and rule based task which also depends on having administrator rights on the GitLab instance to run the necessary scripts. With this in mind, the group decided to create an automated alternative that uses the a teacher's account to achieve the same result. The resulting solution is a set of scripts that retrieves the list of students enrolled in a course from NTNU's

LDAP database and uses the Gitlab API to create the role based hierarchy. Figure 20 highlights the scripts that performs the course creation. This script is available as part of the developed system's template.






Name	Last commit
 .gitignore	add git ignore
 README.md	Update README.md
 config.sh	updated content of config and deploy script
 deploy.sh	updated content of config and deploy script
 submission.sh	psudo kode

Figure 20: Thesis's script for course creation in Gitlab to support private information

The scripts can be downloaded and edited locally by teachers. For example, the necessary variables to create and deploy a subject in Gitlab is defined in "config.sh" on Figure 21.

```
#!/bin/sh

#Token for access to Gitlab-api
Gitlab_token=""

#User password for login@stud.ntnu.no
ntnu_password=""

#Name of subject, effectively becomes root group in Gitlab course hierarchy
subjectCode="DCSG1005"

#Username for login@stud.ntnu.no
username=""

#Amount of tasks projects to be created for students
amountTasks=1

#Name for output file containing all students
studentList="list_of_students.txt"

#Url for Gitlab API requests are sent to
baseUrl="https://gitlab.stud.idi.ntnu.no/api/v4/"
```

Figure 21: config.sh script snippet

Afterwards of updating configuration script "deploy.sh" will retrieve the information as environment variables used throughout the deploy script itself, as displayed on Figure 22.

It is noteworthy that the scripts for creating the course in Gitlab is minimal in its security and


```

sshpass -p $ntnu_passord ssh $brukernavn@login.stud.ntnu.no 'ldapsearch -z -h at.ntnu.no -D "" -b "ou=groups,dc=ntnu,dc=no" "cn=fs_dcsig1005*" | grep
#Retrieve all groups user has from gitlab and store their names and id in groups.txt
curl --location --request GET "${baseUrl}groups?owned=true" --header 'Authorization: Bearer "${gitlab_token}"' | jq '.[] | "\(-name) \(-id)"' | tr
#Read each line in groups and check if the subject code exists already
groups="groups.txt"
while IFS= read -r line;
do
  i=$(echo $line | awk '{print $1}')
  echo "comparing: ${i} : ${emnekode}"
  #Check if the subject code matches retrieved subject groups from gitlab
  if [ $i = $emnekode ]
  then
    #The subject group exists, store its ID
    root_gruppe_id=$(echo $line | awk '{print $2}')
  fi
done < "$groups"

#Check if the subject group existed already
if [ -z "$root_gruppe_id" ];
then

```

Figure 22: deploy.sh script snippet

effectiveness. For example, a multitude of additional scripts or similar solutions would be required for fully implementing all functionalities detailed in the system’s architecture (Section 4.1), which was considered out of scope in this project, considering the thesis is conducted to explore the opportunities instead of developing commercially viable products with adequate testing.

4.4.3 Functionality to publish announcements

A key point going into the project was to make a solid announcement system that would present students with all important events and information in an orderly fashion, but also make an effort to ensure that vital or otherwise urgent messages actually reaches the student body. While the existing announcement system in BlackBoard Learn is well implemented for what it is and does a good job of presenting the information, the only way to directly inform students of newly published announcements is through email. In this project the group has attempted to explore other methods of notifying students whenever an announcement is made, while also learning from what the current system does well.

First off an implementation similar to the existing solution in BlackBoard was made. While ways to deliver announcements to the student body is important, actually presenting these announcements on the website will have to be step one. To allow teachers to create announcements we added a new folder to the structure named *_announcements*. Markdown files in this folder are still found and turned into HTML files by Jekyll, but are completely ignored by the script that creates the index listings. They are instead picked up by a dedicated announcement script that adds them to the top of the index page under a heading that reads ”Announcements”. They

are implemented similarly to BlackBoard Learn's solution, being presented as links that take the user to the full announcement when clicked.

The second method that works side-by-side with the first is an automated system that informs students of changes to the repository through Microsoft Teams. Whenever a new commit is pushed to the repository a web hook is triggered and a post is made to a dedicated Teams channel. This post takes the form of a Teams Card, effectively a small message with fields denoting where changes were made, how many lines were added or changed, and when the change was made. The Card also includes the commit message.

The group believe the best way to notice and perceive an announcement is through channels the target audience already uses. Some professors at NTNU are already using Microsoft Teams for digital lectures and general course coordination. The group thinks building on this Teams usage can make for a great to ensure the announcements actually reach the target audience. Teams offers users the freedom to choose how they are notified by an announcement, including email, push-notifications, or nothing at all.

The group created this feature by a function which sends a curl message to web hook connected to teams. When you update the repository the CI/CD is triggered and a curl message is sent. The curl message contains the changes made to the repository, on teams the changes made to the git-repository is displayed, the layout is similar to the message when using the git pull command.

To configure the functionality Teams needs an incoming web hook. You can create a web hook by navigating to the channel (in a team in Microsoft teams), on the channel you can configure connections. Search "incoming web hook", here you can create a web hook by giving it a name, and when created you receive an open accessible URL. Save the URL in the `_config.yml` (in the Gitlab pages repository) file at the correct variable.

In this functionality the CI/CD pipeline works as follows. first, it finds relevant Gitlab-variables (commit message, date of commit, and so on) and copy them for later use, then a script reads the web hook-URL, a test is ran to ensure a working URL was found (keep in mind that the teams web hook is not a two-way exchange of information, thus if the web hook works there are no way for the CI/CD to know if the message is was transmitted to the correct channel), last the curl message is sent with the variables form the first stage.

4.4.4 Functionality to handle assignments

As of now the developed system does not have the functionalities to support deliveries from students. However, it is possible through extending the user access control already present in construction of course in Gitlab as demonstrated in 4.4.2. Furthermore commands needed have been explored, tested and added to a pseudo code script. The process of which a delivery can be delivered is through Gitlab projects. When a teacher creates the subject Gitlab structure with the auto git structure script, the teacher is given a option on the amount of deliveries processed by the Gitlab structure. The amount of deliveries decides how many folder of Gitlab projects in the project folder, one folder per delivery. Within the a delivery folder (for instance .../project/task1) there is a Gitlab project per student enrolled in the subject. The project is only visible to the student and teacher, other student can't see each others projects. When a deadline arrives a teacher could execute a completed version of the pseudo script. The script should then demote all the students roles on the projects to, for instance guest, the important aspect is that a student is not able to continue editing their project. To demote students should not take long, therefor some what accurately represent a delivery deadline. What may take some time is the process of clone each project, thous this action is made after the students are demote. This is to ensure a student whose project is cloned last gains an advantage by the script using time to clone the projects. When all the projects are cloned the cloned version serves as the students delivery, the projects name is by default the students NTNU username, this will serve as an authentication for the delivery unless the teacher promotes a change where, for instance students write their name(s) in the readme file. After the delivery is made student are promoted back to their original Gitlab role and can continue their project if they desire.

Microsoft Teams Cards

The objective of an announcement is for the announcer to enlighten their target audience. The group believe the best way to notice and perceive an announcement is through the target audiences preferred channels. In this case, students are the audience while a professor is the announcer. During the Covid pandemic some professors use Microsoft Teams for digital lectures. The group think Teams is a great way to transmit an announcement, as teams gives students the freedom to choose how they are notified by an announcement, per e-mail, push-notification

(app), or not.

The group created this feature by a function which sends a *curl* message to webhook connected to teams. As the repository is issuing an update, the CI/CD is triggered and a *curl* message is sent. The message contains the changes made to the repository, which are being later on displayed on a designated Teams channel made for tracking the git-repository. The layout of the Teams message is similar to the message when using the *git pull* command.

To configure the functionality Teams needs an incoming webhook. It is possible to create a webhook by navigating to the channel (in a team in Microsoft teams), where connections can be configured. Furthermore, In channel settings, a webhook can be created by giving it a name, which later on provides the creator with an open accessible URL. Next step of the process is to save the URL in the `_config.yml` (in the Gitlab pages repository) file at the correct variable.

In this functionality the CI/CD pipeline works as follows. Firstly, it finds relevant Gitlab-variables (commit message, date of commit, and so on) and copy them for later use, then a script reads the webhook-url, a test is ran to ensure a working url was found (keep in mind that the teams webhook is not a two-way exchange of information, thus if the webhook works there are no way for the CI/CD to know if the message is was transmitted to the correct channel). Lastly, the *curl* message is sent with the variables form the first stage.

4.4.5 Docker-container

Containers used in the developed system's Gitlab CI/CD pipelines are retrieved from Docker hub [27]. The selection of containers applicable for the developed system or any application relates to the technologies made available and configured on the container image. For example, the developed system utilizes Jekyll application created from the Ruby programming language and therefore needs container images with Ruby installed for running the software. All applications using Ruby commands and interpreters can reuse the same container images and ensure the same environment is used for all deployments of applications. Despite this, during development group members discussed methods for implementing JavaScript vulnerability scanning of docker containers and required additional configuration of Ruby containers. In such occasions one could find unofficial container images on docker hub which are configured for both Ruby and JavaScript but lacking the official stamp of approval from the Docker team. Security

of containers is later discussed in section 4.7.6, but the problem area of having scalable and maintainable container images is to be discussed further at present.

Users of the developed system can create their own container images including all dependencies from setup or make use of base images to be configured during Runtime to install dependencies instead [28]. In practice, but solutions result in the container images with appropriate configurations for the developed system, but the difference lays within the approach of maintaining of images. Furthermore, using a base image and installing required dependencies enables developers to work from the same foundation to work from. On the other hand, administrators would inherently increase the risks of having different environments throughout their pipelines as each job requires their own designated installations steps. In contrast, developing a container image with all dependencies part of the base image alleviates developers from configuring the images themselves during pipeline development. In spite of this, base container images would frequently require updates as dependencies introduce vulnerabilities or general changes.

All in all, the developed system utilizes container images from the publicly available Docker hub. The approach for retrieving container images in use has its up and downsides. With regard to thesis, group members have concluded that teachers should take use of base images included in the developed system's template or incorporate other official base images when needed.

4.4.6 Timing-issues

Pipelines in the developed system has timing-issues during web testing. Whenever a pipeline is triggered, for example by uploading new markdown documents, Gitlab CI/CD starts the pipeline stages as defined in “.gitlab-ci.yaml” file. With regard to the developed system, several steps included in the established pipeline executes web based tests to report on the functionality of transformed websites. Ideally, transformed web pages should not be deployed if web testing stages reports errors. Despite this, pipeline stages for rendering pages are inconsistent in time before updating the publically available pages. As a consequence, web based tests in the current iteration of developed system has a risk of reporting on preceding transformed web pages if new changes to pages are not rendered and uploaded before the tests are ran. It is noteworthy that stages in template pipeline is configured to be ran after each others completion sequentially, unfortunately the “pages” job part of “deployment” stage is a Gitlab service that group members

cannot configure themselves and reports itself as completed before new pages are uploaded. Considering the scope of thesis, group members concluded with the timing-issue being an issue for future improvement.

To solve the timing the separate containers for web testing and rendering pages could be configured to share the same network environment and report on locally rendered pages before continuing the pipeline. To further explain the differences, current configuration of web testing are ran on a Cypress container requesting the publically available web pages hosted on the Gitlab instance, for example *https://gitlab.stud.idi.ntnu.no/*. In contrast, the proposed solution have a container host the rendered pages locally with shared network resources to web testing containers that run tests on *http://localhost:8080/*.

4.4.7 Deploy stage in Gitlab CI

The ultimate stage of the GitLab CI pipeline runs a series of scripts to prepare the file structure in the GitLab repository for being rendered as a website through Jekyll. This is necessary because while the final website is completely static, it still has to reflect the original repository. There are therefore several elements that need to be generated based on the file structure of the repository that would be impractical for the teacher to do manually.

Generate sidenav and nav pages

The first of these scripts is responsible for generating the side navigation bar as well as static HTML sites for every possible navigation page, the latter of which will be explained shortly. The side navigation bar is generated based on the directories directly below the `_content` directory visible in 23, excluding the predefined directories “*announcements*” and “*nav*”. The entries in the side navigation bar link directly to the navigation pages representing their respective directories.

Every sub-directory of the “content” directory generates a respective navigation page exemplified in 24. Each navigation page contains links to both the directory’s own files and sub-directories. Each directory is turned into a markdown file and stored in the “content/nav” directory with contents as seen in 25. The markdown files are named based on the entire path to

the relevant directory as well as an added “-page” at the end, in order to ensure that each and every directory is created with a unique filename. E.g. a hypothetical folder “*content/reading-material/extra*” would be given the name “*reading-material-extra-page.md*”.

The markdown files for the navigation pages are generated before Jekyll runs and are therefore automatically turned into HTML pages just like any other markdown file in the “content” folder. Jekyll also assigns the name of the source markdown file as the title of the page by default, which means the nav pages would have ugly names not fit for the front-end of the system. To circumvent this the script sets the front matter of the markdown file to include the “title” field, assigning to this the name of the current directory. This allows for several nav pages to have the same title while avoiding them having the same filename.

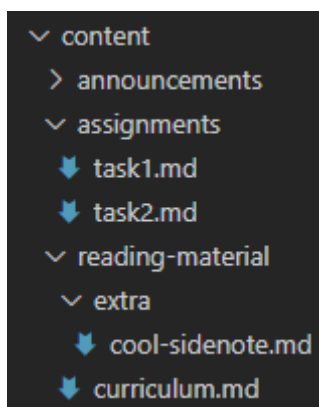


Figure 23: Source file structure

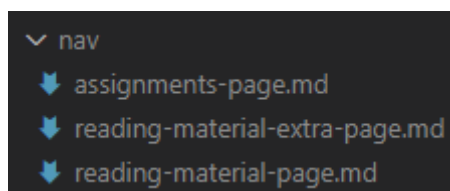


Figure 24: Generated nav directory

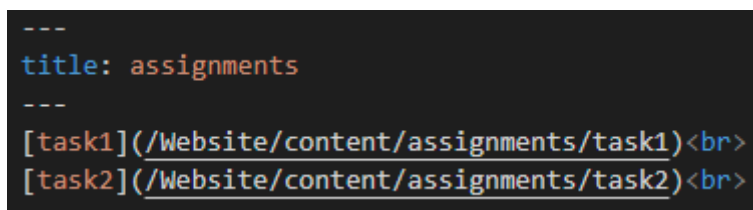


Figure 25: Generated nav page markdown file

Move content

The next step in the deploying process is to move the structure of the *content* folder into the *webpage content* folder. As a brief reminder, the base content folder is the one teachers are meant to interact with, while the webpage content folder is where Jekyll builds the site from. Distancing these two from each other allows end users to use the system without knowing the ins and outs of how exactly Jekyll works as the system will translate an ordinary file structure into a Jekyll appropriate one.

To support users who know Jekyll and want to define their own structure without any help from the system the first step of the content moving script is to check whether the file structure of the content folder already fulfills Jekyll's requirements. The key requirement that is being looked for is the obligatory presence of a *_posts* directory. Should one or more such directories be found the system will simply move the contents into the webpage folder as is, making no further changes.

In the event that the content structure is not Jekyll compatible the system will make a few minor changes. Each sub-directory in the original content folder will be recreated as an empty directory in the webpage content directory. If the original directory contains one or more non-directory files a *_posts* folder will be created and the files copied over into this new *_posts* folder. The end result is a webpage content folder that is structurally identical to the original content folder with the exception of putting all non-directory files into *_posts* directories, allowing Jekyll to find and work with them. An example source directory and resulting website directory can be seen in Figure 26.

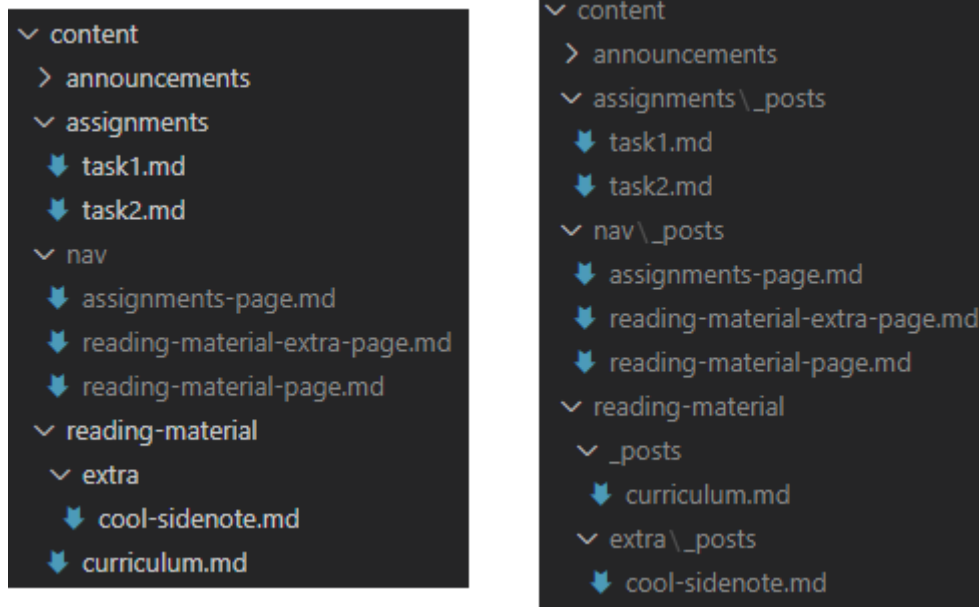


Figure 26: Source content folder (left) and resulting website folder (right).

Make index table of contents

The content listings on the index page of the site has the purpose of giving a complete overview of all the course’s contents for student users. In short this is accomplished by recursing through each directory in the content folder and creating an HTML file with a list links to each directory’s navigation page and every markdown file’s generated HTML page. The logic for retrieving lists of files and directories within a given directory are reused several times throughout the scripts and can be quite confusing to read unless the reader is sufficiently used to UNIX commands like `grep`. The complete command used for this operation is as follows:

```
# Get names of all directories
directories=$(ls -l $1 | grep '^d' | grep -o '[^ ]*$' | tr -d '/')
```

Figure 27: Command retrieving list of directories

The command seen in Figure 27 performs a series of operations in order to return a list of the names of every directory contained within the current directory and storing this list as an array in the variable “directories”. The “|” sign denotes the piping of the result of the previous operation into the next operation, essentially running the initial input through an “assembly line” of operations to achieve the desired result.

- `ls -l $1`

ls Lists content of the directory specified in the variable \$1.

-l Option for ls that prints more information about every item in the list. Used here to determine the type of file.

\$1 The first argument passed into the function containing this line of code. In this case the path to the current directory of the recursion.

- `grep '^d'`

grep Returns any lines matching the given pattern.

pattern The '^d' pattern matches any line that starts with the character 'd'. `ls -l` output includes among other information a character denoting the type of file at the start of the line. The character for directories is 'd'.

- `grep -o '[^]*$'`

-o Option for grep that returns only the exact part of the line that matches the given pattern.

pattern The '[^]*\$' pattern matches a continuous string of non-space characters next to the end of the line, i.e. the last word of any given line. This is used to remove unnecessary data from the `ls -l` operation, leaving only the name of the directory.

- `tr -d '/'`

tr Command used to replace one character for another in a text.

-d Option for tr that removes a given character instead of replacing it with something else. Used here to remove the trailing '/' after the directories' names.

4.5 LMS installation and deployment

The developed system is logically separated into three different projects interconnected through Gitlab CI pipeline.

Before reading, the developed system utilizes Snyk[23] a vulnerability scanning tool later discussed in 4.7.7 for its security. To use the developed system as intended, users are required to

create a Snyk user profile and create a token for applications such as the developed system to utilize.

Firstly, a project to create and publish websites as Gitlab Pages. To install and configure Gitlab Pages fork the working repository, in case it is inaccessible refer to the latter repository:

Working repositories: <https://gitlab.stud.idi.ntnu.no/bachelor1>

Code repository: <https://git.gvk.idi.ntnu.no/Fabian/dcsg2900-gitlab-pages>

The working repository has been configured with a CI variable in its project setting, and additionally uses downstream projects[29] as part of its pipeline. Furthermore, unit testing of the pages are done with a separate Cypress project, and system testing of WCAG compliance is reported using an GitLab automated reporting project.

Configuring the CI variable can be done through the GitLab web interface. First open the “Settings” menu, thereafter “CI” menu and the “Variables” section can be configured. Create a variable named “SNYK_TOKEN” and give it the appropriate value retrieved from user’s Snyk profile.

Downstream projects are defined as part of stages in the Gitlab Pages project’s “.gitlab-ci.yml” file. For example, there are currently five active stages:

- vuln-scan
- deploy
- web-testing
- web-accessibility-report
- notification

web-testing and web-accessibility-report are intended to be separate projects that can be interchanged with similar functionalities depending on the user’s wishes. In practice, users need to withdraw the correspondingly named folders from the code repository and create them as separate projects. Thereafter update the Gitlab Pages project to trigger the aforementioned web-testing and web-accessibility-report projects. Figure 28 display a snippet of the .gitlab-ci.yml where the connection between the file and separate projects is made.

```
#Multi project CI - test cypress and p11ly on our gitlab pages
cypress:
  stage: web-testing #which stage of CI/CD pipeline this job is ran
  variables:
    url: $CI_PAGES_URL
  trigger:
    project:
      - bachelor1/cypress_gitlab_pages #Run cypress our own defined tests against gitlab pages. NB! does not test
#Multi project CI - test p11ly on our gitlab pages for web assesibility testing reports
pally:
  stage: web-accessibility-report #which stage of CI/CD pipeline this job is ran
  variables:
    ally_urls: $CI_PAGES_URL
  trigger:
    project:
      - bachelor1/webpage-accessibility-test #Run p11ly's pre defined assesibility tests on our gitlab pages
```

Figure 28: Multi-project pipeline configuration

Course specific configuration is only necessary for the Gitlab Pages project. As a baseline, the group members recommend following the installation guide provided in the code repository's "Readme.md" file.

4.5.1 Prerequisites

The group assume the client already has a Gitlab infrastructure available. Otherwise, the client should follow GitLab's online documentation for how to set up the platform [30], before continuing to create the course. Besides the platform, it is required that the client has the following:

- Jekyll Service to convert markdown to HTML.
- Docker Service installed, as the Jekyll client requires it.
- Ruby

4.6 File Structure Overview

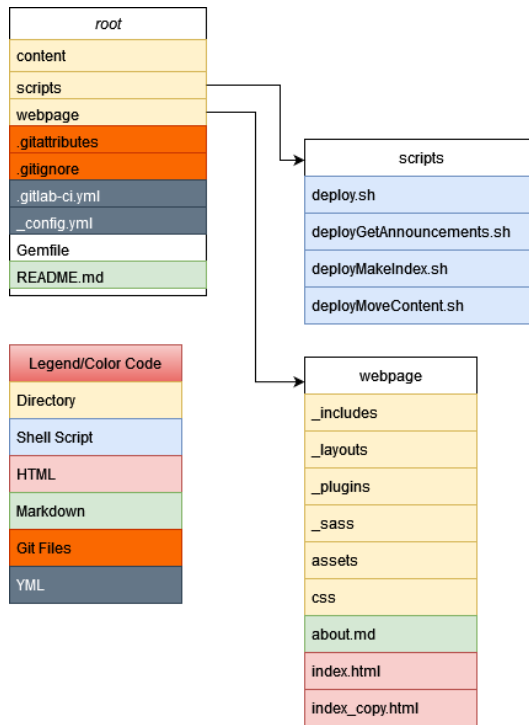


Figure 29: File structure of the GitLab repository.

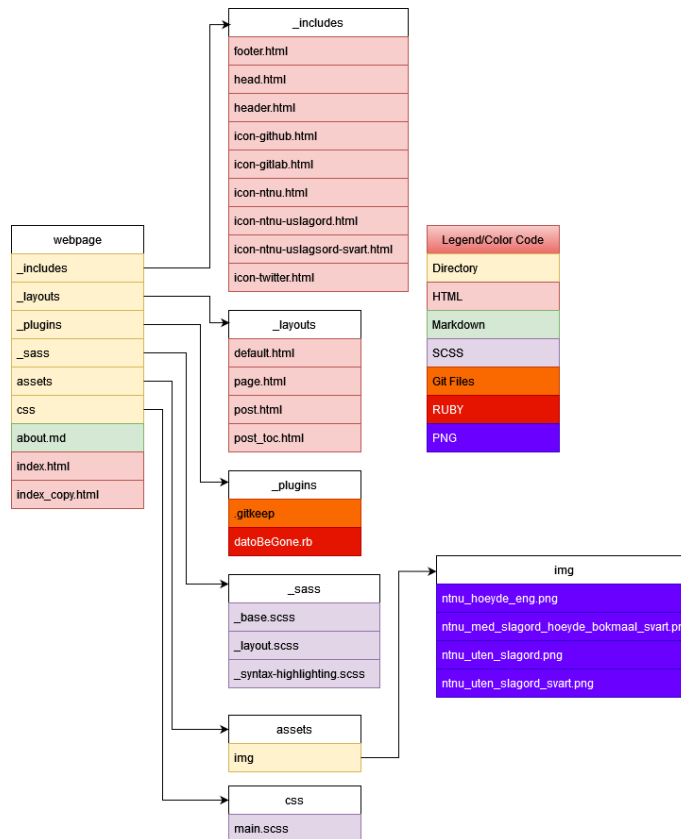


Figure 30: File structure of the `/webpage` directory within the GitLab repository.

4.7 Security

The intention of secure design is a resulting system which enforces imperative security measures. For instance, systems that enact authentication, accountability, authorization, data integrity and availability is in thesis considered to be secure. As a baseline, all aspects of the developed system is created to be easily configurable and changeable depending on the end-user's requirements. For example, the majority of security measures included in the template project are part of CI/CD pipeline in which users can remove existing or add additional steps. With this in mind, the developed system's security measures exists as an example for the minimum viable product as defined by the group and client. Additionally, the group considers any system to be prone to implementation defects and believes further research and development of security measures is required for a commercially viable product. The existing procedures have been designed to reduce available endpoints for exploitation by threat actors and is considered adequately secure for the scope set for the project.

The following sections aim to describe what procedures are part of the developed systems. Firstly, procedures for endpoint protection on a platform level will be presented. Secondly, security measures part of the platform itself. Lastly, security of the underlying technologies is described as well.

4.7.1 Endpoint Protection

After discussion with Henrik Johnsen, administrator of the Faculty of Informatics GitLab Instance, the group was informed that the relevant GitLab instance is protected by Cisco Advanced Malware Protection (AMP). This is a Endpoint Detection and Response (EDR) tool that is suited to scan the internal system and respond to any potential threats [31]. It differs from a standard antivirus software by having a broader scope of action: An EDR system provides not only antivirus functionality, but also functions as a firewall, as well as offering tools for monitoring and providing access control [32]. Furthermore, an EDR system can delve more deeply into the threat and provide additional insight as opposing to an antivirus which simply removes it. The additional info an EDR system can provide, can be useful for further investigations resolving around an incident.

4.7.2 GitLab's Security Measures

GitLab provides a couple security measures on its own, further minimizing the attack surface. The following paragraphs will describe a handful of the implementations, which are especially relevant for our product, besides being beneficial for the platform altogether.

Password Policies

In terms of password storage and requirements enforcement, GitLab Has numerous implementations at hand, to ensure the confidentiality of user's credentials within the database. All stored passwords have these three actions performed on them: GitLab uses *bcrypt()* for Hashing user passwords. As the function description states, it accepts three inputs: the password itself, the input cost, and a salt value. As the documentation states, all salt values are randomly generated and unique to every password. The uniqueness of the salt values is to minimize the compromise in case of a threat actor that has successfully discovered a salt value. The output from *bcrypt* is a hash value, representing the password. As a final step of password obscuring before storing in the database, the password is "stretched" by running the hashing process anew for several hundred times [33]. The output of the stretching process is the hash which is being stored in the GitLab database. Upon user login, the input hash value is compared with the stored hash. Should the values be alike, the user access is granted.

This measure protects the database against brute-force attacks and techniques alike, where the attacker tests self-made password hashes against the database-stored. However, it does not protect against the human error of "handing the password over", as is often the case with Phishing attacks. As a counter-measure on successful login attempts by threat actors, GitLab has enforced Two-factor authentication for users. This design principle is also called "Separation of Privilege", where the user requires several methods of authentication in order to bypass the system [34].

User file Uploads

To create a separation between public and private information, we created a structure of groups, subgroups and projects with different access control, similar to mariusz's structure. A part of the structure is a group, hand-ins, students are given a project in this group for each assignment through our service. The projects are private to the student, as the teacher has root access to the structure and are the git owner of the projects they also have access. When a submission deadline arrives a script can automatically demote a students project membership type so the student can push new code, this goes fast. After all students are demoted the script will start cloning the projects, the cloning process can take some time, therefore it is important to demote all student so as no student can push new code while other projects are being cloned, it should not favor a student for processing their project last.

The group believe our submission service to be safe, as each project is isolated and not accessibly to other students and no one other then students in the subject are given a project(s).

However as of now it does require a lot of manual labour as there are no system for recording who passed and how failed. A professor can give feedback through for instance, the readme file or a projects wiki page, the feedback can say if the student(s) passed or not but the professor do not posses, if not created by him/her self, a centralized system for his/her students results. As of now the submission script is only in pseudo code, but all the lines of code needed exists and are tested.

The current project based submissions are best suited for Git based projects (usually programming projects), but can be used to deliver a single file. An upload button on the website is the simplest solution for a student. However a file upload function may create some problems, for instance, is or can the file be executed, how do we know which file is owned by who and can students read each other delivered submission?

On a static website there are no way to know who a submission belong to, unless the submitter has written their name on their submission. To automate a perfect submission function the website need to be to some degree dynamic, contain a database and have a detection system where only students are allowed to deliver a task. The last one is currently a problem, the current configuration of IDI's Gitlab instance has no way to "close" or make private a Gitlab

page. Thus all Gitlab pages are always public, meaning if an upload button is on a Gitlab page nothing is hindering whoever from submitting their files. A script can compare NTNU LDAP's list of student in a subject to the submitted file's content, to see if the submitter has written their name in the file and is taking the subject. The problem is at such a script will make a couple of request to NTNU's servers and need computer power to potentially compare a whole document per submission which stacks up fast.

Announcements Teams Webhooks

Part of the Gitlab CI/CD pipeline is the notification stage with the purpose of sending announcements to the institution's preferred communication channel. For example, in our thesis we have utilized the Teams integration "incoming webhook" that establishes a unique URL accessible online that forwards web-requests into a dedicated Teams channel [35]. It is noteworthy that the application does not authenticate requests to its URL and a malicious user could send misleading messages. Lack of authentication proves a vital flaw with the existing solution with Teams, and it is recommended to develop a similar application that validates users before publishing messages. The content of messages themselves is not considered a security threat as "incoming webhooks" uses declarative message structure which prevents insertion of malicious code.

Runner Security

As previously discussed in "4.1.1. Overview Flowchart", as the user issues a "Git Push", a process also known as "Gitlab runner" executes a series of jobs within a pipeline, with the aforementioned containing lines of code to run. As it stands, the stud.idi.ntnu GitLab instance has 4 shared runners[36], all of which are accessible for job handling for any GitLab repository. Alternatively, users can create *their own* runners, which consequentially have to be manually updated by their creator, but are also open for tailoring to the creator's desire. Security complexities arise when we take a look at the usage of said runners: As it goes for the latter runner alternative, the dangers of a compromised runner host can be used for executing malicious commands/programs, exposing the machine's working environment for further attacks or the repository altogether, should the information be considered confidential. Considering that self-made runners can also be used in different repositories, keeping them secured is vital for the security

of the projects.

Specified in the Security section of GitLab, a best practice to reduce the extent of damage applicable upon compromise is to make the runners more “ephemeral”, resulting in a shorter lifespan of a single instance [37]. Instead of keeping a single iteration running for several jobs, it is advisable to create a new instance for every running job. This isolates the runner from additional tasks, and reduces the attack surface of the instance upon compromise.

An additional detail to keep in mind when configuring runners is their privilege in the system. For example, in development there are stages with dind binding (docker-in-docker) that evaluates and analyzes the quality of code in a given repository. Running containers in privileged mode exposes the system to a multitude of vulnerability issues as the container is no longer residing within an isolated environment and exposes the kernel and external hardware resources of hosting computer.

Running a CI/CD Job in privileged mode could be beneficial should a desired application require connection with the host, as privileged mode allows actions outside of the enclosed environment. However, this also amplifies the risk of being compromised, as the attacker. Documentation advises runners operating with elevated privileges to reside within isolated environments, as well as being in ephemeral state, meaning the machine should be terminated as soon as its initial task is completed.

CI/CD Variables

Variables empower developers to create dynamic processes. Simply put, variables are an abstract storage location paired with an associated symbolic name which contains a value [38]. Logic can also be applied to variables, creating branches of possible outcomes depending on the value stored within a variable. With regards to the developed system, Gitlab CI/CD variables are used in pipelines for configuration and dynamic scripting. There are four different use cases of Gitlab CI/CD variables in the developed system:

- Variables in the “.gitlab-ci.yml” file
- Project variables

-
- Multi-project variables
 - Predefined variables

Variables in the “.gitlab-ci.yml” file are defined with the “variable” keyword accessible globally throughout all jobs or isolated within a single job instance.

Project variables are defined in a project’s CI/CD settings under the “Variables” section. A unique attribute of such variables is they can only be accessed within the project and can be configured to be masked in scripted jobs. For example, project variables can be used to store critical data for authentication such as password and is used to store login information for Snyk monitoring elaborated on in section 4.7.7.

Multi project variables are part of multi project pipelines [39] that span over different projects. For example, the developed system consists of multiple projects in which “Gitlab pages” triggers other projects for testing purposes. Moreover, Upstream job define variables and passes their values into Downstream job [29].

Predefined variables are available in any jobs and contains information concerning the git based system. To be specific, the developed system incorporates committed messages and name of projects in different pipelines for dynamic purposes.

4.7.3 GitLab Access Control

An important security aspect of protecting a GitLab repository is to restrict access to unauthorized persons. This is possible with the use of two methods: Role permission limitation, or by using *access tokens*. The former method is available in the “Members” tab when viewing the repository settings. When inviting new members, it is possible to regulate their access as defined by five tiers: Guest, Reporter, Developer, Maintainer and Owner, each providing the user with more permissions than the previous [40].

The latter option, access token control, is an alternative solution for the same challenge, albeit with another use case: Whereas account-based project access is regulating human access, token-based access control can be useful for automated, non-human processes such as scripts and applications. Furthermore, tokens work in the same way as passwords, by being provided

alongside the request to the respective repository for authentication [41]. These tools, however, will protect the repository only if the people or processes are authenticated with the least privilege required. It will correctly protect the resource if it is correctly implemented.

Open/closed Gitlab pages

By default all Gitlab pages are public regardless of the projects accessibility. Access control for Gitlab pages can be activated, if activated only members (at least Guest-members) can access the web site. It is important to note that the access control is active in the project setting and project wide, meaning access is either public or closed for all web pages in a website generated by Gitlab pages. There is not possible to have certain pages be closed while others are public, it is however possible to have two web site where one i open while the other is closed. If an unauthenticated user try to access the website they are redirected to Gitlab, if authentication is successful the user are redirected back to the website. See the source of this paragraph for how to configure Gitlab pages access control [42].

In this project we used NTNU's instance of Gitlab, `gitlab.stud.idi.ntnu.no`, this instance does not support closed Gitlab pages. To give Gitlab pages access control, pages has to be registered as an Open Authorization (OAuth) application with Gitlab. To enable gitlab pages access control add the line `"gitlab_pages['access_control'] = true"` to the Gitlab configuration file (`/etc/gitlab/gitlab.rb`) [43].

4.7.4 Ruby

Ruby, being an open-source project, allows its users and and other persons of interest to inspect the code. This goes well in hand with Ruby's Bug Bounty program on HackerOne. The program offers a monetary reward to anyone who finds a vulnerability in their software.

On the topic of response time after submitted vulnerability, the HackerOne website provides statistics on "Average time to first response" and "Average time to resolution". These times are listed at 4 days and 3 months, respectively [44].

Many interpreted languages (such as Python, Ruby, and JavaScript) have an eval function that consumes a string consisting of syntax in that language and invokes the language's interpreter on the string. Use of a language's eval facility can permit the implementation of very powerful

features with little code, and is therefore tempting. It is also very dangerous. If attackers can influence even part of a string that is evaluated and that substring is not appropriately validated or encoded, they can often execute arbitrary code as a result [45].

4.7.5 Jekyll

The CVE database for Jekyll-related vulnerabilities reveals that only a single vulnerability has ever been documented, back in 2018. Named CVE-2018-17567, the vulnerability allowed a threat actor to access confidential files by specifying a symlink in the configuration file [46].

Systems consuming complex data formats (such as XML documents, image file formats, or word processing file formats) might perform parsing, syntactic validation, and semantic validation of input files in a dedicated validation module whose output is a validated internal object representation of the input document. Parsers and validators must themselves be designed to robustly cope with potentially malicious or malformed inputs.

Static versus dynamic websites

The information-related challenges were solved with the use of static websites, as our client desired. However, one may wonder how the layout would look like if there were no limitations on website format.

Static websites, as the name implies, are unchangeable: No matter the user, no matter the conditions met, the website will look the same. Consequentially, a static website is more lightweight in complexity: These websites are made with no more than HTML and CSS. Its counterpart, the dynamic website, is far more developed, both in its look and in its possibilities. As opposed to the former, a dynamic website is open for interactivity with the visitor, and can change its look or functionality depending on the user and their respective privileges on the system [47]. Additionally, the websites are also suited to interact with additional infrastructural components, such as a database.

However, with a broadened list of supported technologies, comes a broadened attack surface. As GitLab pages are considered static websites, there are no entry points for an attacker to utilize. The surface increases as the website implements additional components: Respectively, if the

connected database does not have input validation, it is theoretically possible for an attacker to manipulate the overlooked detail to their own malicious favor [48].

4.7.6 Container Security

Isolation of processes, separation of services and control groups contributes to making containers secure. As described in Section 2.7, containers are isolated and can only view resources in the given namespace. Availability of detectable resources are limited by control groups. In fact, containers image specification and runtime specifications are fully configurable in which resources such as memory can be completely prohibited if needed. Deep diving into different configurations of containers and researching the resulting effects is out of scope for thesis. Moreover, thesis's developed system is hosted on a Gitlab platform hosted at NTNU with its own administration. Group members have been in contact with, as of writing, the current administrator Henrik Johnsen [49] to discuss possible methods to test different rulesets to apply to the platform. Henrik Johnsen was receptive to any suggestion on configuring the platform with proper justification and documentation (see Appendix C.5). Influencing platform configuration is marked as a future potential for the developed system and considering the scope of project ended there. It is note worthy that docker ensures container isolation with creating namespaces as a default. In addition, enabling USER namespace maps privileged users in containers to non privileged users on host machine. Privileged operations, such as terminating other processes or reading and writing to root file system cannot influence the host.

Exploitations arising from inadequate or insufficient usage of namespaces and control groups is exemplified in the research article *Houdini's Escape: Breaking the Resource Rein of Linux Control Groups* presented on ACM SIGSAC Conference on Computer and Communications Security. The article elaborates on multiple attack vectors that potentially breaches the limitations enforced by control groups. The central theme is when child tasks are created from parent processes they inherit recursively all rule sets as its parent task. However, when the newly spawned process generates workloads on processes not affected by initiating control groups a de-association occurs in which attackers gain leverage of the system [50]. Similar cases can result in denial of service through generating excessive tasks for the underlying system preventing applications to gain access to essential computer resources.

To summarize, containers are inherently secure from limitations put on namespaces and control groups. Despite this, any security measures are rendered useless without proper configuration and administration. End-users should therefore thoroughly analyse that correct handling of namespaces and control groups are in adherence to their standards and wishes. After all, the developed system is created as a template for end-users to customize depending on their needs and hosted on interchangeable cloud platforms. As a rule of thumb, group members recommend allocating specific or trusted shared resources on each individual namespaces containers become part of or inherits from. For example, end-users with control of the hosting platform should create overarching namespaces and a set of user groups with rules applied from control groups for containers. Furthermore, containers should automatically inherit configured namespaces and enroll as an user of the user group when launching.

4.7.7 Vulnerability scanning

Any systems is prone to errors. To prevent problems arising from dependencies the vulnerability scanning tool Snyk (pronounced “sneak” [51]) is used to report common security issues part of the developed system. Snyk services are acknowledged for their insights by several leading service providers such as Amazon, Docker and Google. In fact, all mentioned parties and more have partnered with Snyk and integrated their services as part of their platforms [52]. To further exemplify Snyk’s vulnerability scanning tool to be considered as an expert opinions the following citation is given:

“Cloud native development has become a critical advantage to organizations looking to deliver modern products to market more efficiently. By bringing Snyk’s vulnerability insights into the new Amazon Inspector, we’re enabling security teams to leverage truly comprehensive, contextual vulnerability information that helps prioritize the most severe vulnerabilities first and further empowers agile software development on AWS.”

- Michael Fuller, Director of Product Management for AWS Security Services [53]

Snyk offers scanning of open-source dependencies, one’s own code, containers and infrastructure as code. In relation to the thesis, Snyk is used to scan container deploying web pages and

defined in the CI/CD pipeline, as seen in Figure 31.

```
###
# auth: Jeff Mclean
# source: https://github.com/snyk-labs/snyk-cicd-integration-examples/blob/master/GitLabCICD/gitlab-npm.yml
###
dependency_scanning:
  stage: vuln-scan
  variables:
    SNYK_TOKEN: $SNYK_TOKEN
  script:
    - apt-get update
    - apt-get -y install nodejs
    - apt-get -y install npm
    - bundle install
    # Install npm, snyk, and snyk-to-html
    - npm install -g npm@latest
    - npm install -g snyk
    - npm install snyk-to-html -g
    # Run snyk help, snyk auth, snyk monitor, snyk test to break build and out report
    - snyk --help
    - snyk auth $SNYK_TOKEN
    - snyk monitor --project-name=$CI_PROJECT_NAME
    - snyk test --json | snyk-to-html -o snyk_results.html
  artifacts: # Save report to artifacts
    when: always
    paths:
      - snyk_results.html
```

Figure 31: Gitlab CI/CD Snyk configuration

The scanning tool searches through all dependencies listed in the developed system’s gemfile, a format for describing gem dependencies for Ruby programs [54]. Vulnerability entries for each dependency is searched for in Snyk’s own databses and a resulting HTML report is generated. A thorough description of the current configuration of Snyk in Gitlab CI/CD is described as follows:

- Retrieves Gitlab CI/CD variable “SNYK_TOKEN” to authenticate against Snyk user profile
- Installs Node.js and Node Packet Manager
- Installs dependencies
- Installs Snyk
- Locates file containing dependencies.
- Searches Snyk’s databases for vulnerabilities related to dependencies.
- Creates report viewable on Snyk user profile
- Creates HTML report listing all vulnerabilities with remedying actions.
- Upload HTML report as artifact

In Snyk’s own web interface, end-users are enabled to view all reports generated to that user profile, as displayed on Figure 32. Multiple projects can be connected to the same profile. Image below depicts the history of the developed system in which two dependencies generated two security issues that were addresses as a result of Snyk’s reporting.



Figure 32: Snyk vulnerability history

Vulnerabilities are presented individually in the report and lists the relevant information, namely source of origin and remedying actions. For example, one of the aforementioned security issues showcased was solved through updating the dependency, as displayed on Figure 33:

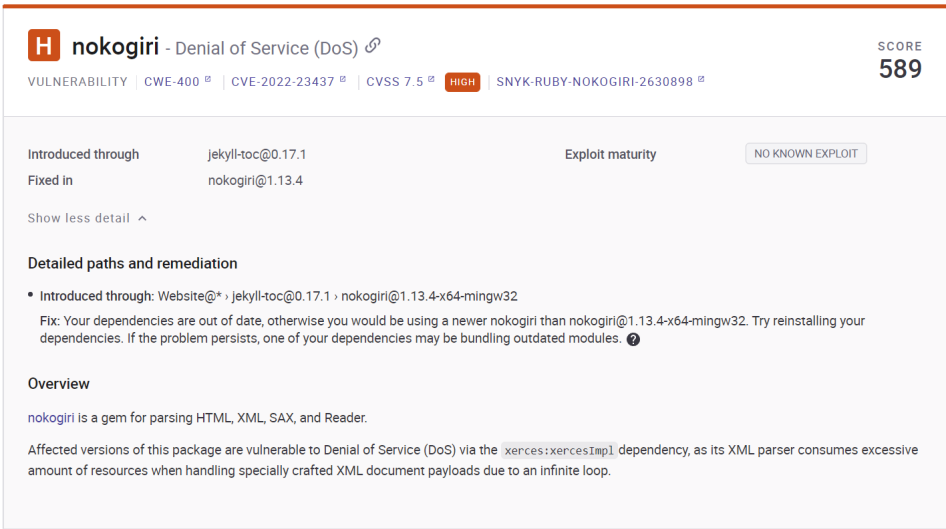


Figure 33: Snyk issue description

Figure 34 displays an excerpt from HTML reports, which listed a total of zero vulnerabilities found from the developed system’s dependencies.

4.7.8 Trusted Docker images

Docker images used in the developed system are retrieved from the public repository service Docker hub [27]. Users are enabled to freely upload and store images onto the service. As a

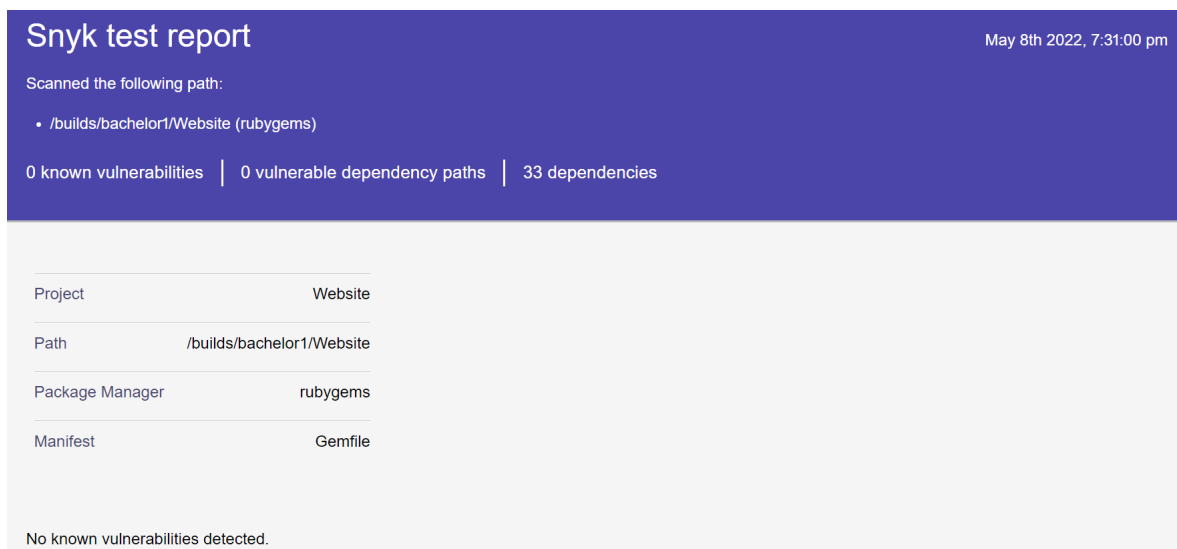


Figure 34: Snyk report

consequence, requesting said containers to be used for applications without caution can lead to vulnerabilities issues. Official distributions are therefore marked with a label to counteract such problems. The label depicts that the container image is curated and approved by Docker [55]. With this in mind, the developed system uses the official and trusted container image "Ruby 3:0" [56]. Figure 35 is the official ruby Docker image.

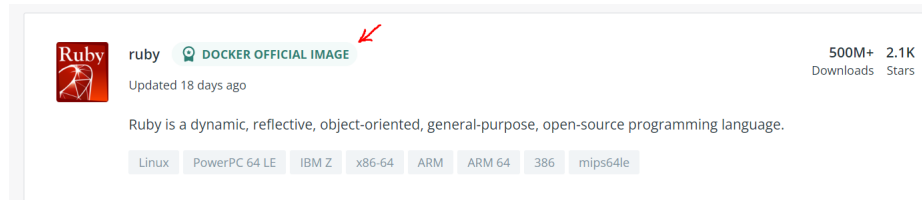


Figure 35: Docker official images

It is of importance that the developed system's project template does include an unofficial image to perform web testing, namely "cypress/included". Cypress is used to exemplify means of conducting web testing in CI/CD pipeline and is interchangeable of similar web testing frameworks. Despite this, the images supplied from Cypress has a combined total of over one hundred million downloads which group members believed would have gained traction if security issues were found. In addition, Snyk reports on container image has resulted in zero dependency vulnerabilities.

Lastly, the container image used for testing WCAG [57] compliance of the developed system's pages is part of Gitlab report type "accessibility" and is therefore considered as trusted [58].

4.7.9 CI/CD best practices

CI and CD depicts two practices for frequently and quickly deliver applications to customers. Alongside the growth of the tech industry a set of standards forms and is called best practices. With regard to security, best practices often includes several actions for the purpose of improving confidence in quality and integrity of software. Furthermore, CI/CD achieving confidence and credibility in the developed system is done with automated testing, a best practice of the Development & Operations industry.

As a result of automated testing, users are enabled rapid development and can monitoring each part of the cycle. To exemplify, history of pipelines are made available from GitLab and can be used to draw statistics for tracking speed of deployments over time, identifying recurring problem areas when pipelines fail and comparing different iterations of the developed system. To summarize, automated testing is a best practice of CI/CD and can be used for system feedback and users gain confidence in the software.

4.8 Quality Assurance

In order to ensure the best possible quality of our end result, we have tried to adhere to good practices of quality assurance. This section is set to describe what measures have been implemented to ensure a clear visibility on improvement of the solution.

4.8.1 Definition of Requirements

To make sure the entire team was on the same page about what exactly the functionalities we were working on at a given time were, we started every sprint by creating a *Definition of Done* document. The document stated the goals of the group, which were due to completion at the end of their dedicated sprint. This also functioned as a checklist at the end of the sprint, that helped with making sprint reviews, allowing us to go through the list and make sure everything was implemented the way we envisioned. As is natural in an agile work environment, these requirements would sometimes change underway, usually to include a dependency or some other unforeseen challenge.

The key parts of these Definitions of Done are *Assumptions and Dependencies* and *Functional Requirements*. Assumptions and Dependencies serve as documentation for the assumptions we made before we defined the requirements for the functionality, as well as any other technologies that are required for it to work. Assumptions usually serve to limit the scope of the functionality by excluding fringe or otherwise unnecessary use cases in order to make the development process more efficient. Functional Requirements stated concise and measured prerequisites, which are deemed necessary in order to complete the sprint tasks.

4.8.2 Automated Reports

Gitlab offers a wide range of “Auto DevOps[59]” features which is a collection of pre-configured docker containers that work together to support the software delivery process [59]. For instance, the Auto DevOps features used in our system creates reports on accessibility of Gitlab Pages and quality of code in repository.

Automated reports are part of the Gitlab CI/CD that triggers whenever a change is made to the system. The reports are made available as “GitLab Job artifact” , an archive of folders or documents in relation to a job in a given Gitlab CI/CD. For example, “Pa11y” is a free and open source tool part of Auto DevOps for measuring the accessibility of web sites based on WCAG 2.1 rules [57] configured to analyse web sites made available through our Gitlab Pages with a resulting report named “accessibility”. Pa11y report example can be seen in Figure 36.

Report entails specific description of the WCAG rule the implementation is violating and remedying instructions. Modern websites should adhere to standards set by WCAG as their specifications lead to a unifying design which benefits users with disabilities. Especially in an academical setting institutions are imposed to make their studies available to all students. End users of our system are enabled to improve their courses digital front by reading this report, or taking advantage of the template course part of thesis which has a resulting report of zero errors.

Users interested in further developing the system should make the most of all reports. Moreover, CodeQuality ensures the code in a repository is readable, uncomplicated and vacant of common vulnerability issues. Figure 37 displays the structure of the provided reports after a singular scan.

Accessibility Report For "https://bachelor1.pages.stud.idi.ntnu.no/Website/"

Generated at: Tue Apr 26 2022 10:37:12 GMT+0000 (Coordinated Universal Time)

21 errors 30 warnings 0 notices

Error: The html element should have a lang or xml:lang attribute which describes the language of the document.

WCAG2AA.Principle3.Guideline3_1.3_1_1.H57.2

```
<html><head>
  <meta charset="utf-8">...</html> (select with "html")
```

Warning: Interfering with a user agent's ability to zoom may be a failure of this Success Criterion.

WCAG2AA.Principle1.Guideline1_4.1_4_10.C32,C31,C33,C38,SCR34,G206

```
<meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1.0,
user-scalable=no"> (select with "html > head > meta:nth-child(3)")
```

Warning: This element has "position: fixed". This may require scrolling in two dimensions, which is considered a failure of this Success Criterion.

WCAG2AA.Principle1.Guideline1_4.1_4_10.C32,C31,C33,C38,SCR34,G206

```
<div class="navbar" id="navbar">
```

Figure 36: Pally web accessibility report

The screenshot shows a 'Issues' section with a 'TODO found' category. It lists three issues:

- Issue 1: A TODO comment in README.md regarding markdown file naming. The code snippet shows:

```
79
80 Markdown files to become websites needs to be named ".md" as of now because of "datoBeGone" plug
81 TODO: make markdown extension all encompassing
82
83 ## Directory/folder structure
```
- Issue 2: A Ruby code snippet where an 'end' tag is not aligned with a 'class' tag. The code snippet shows:

```
11 read_publishable(dir, "_drafts", /*\md$|.*\markdown$/)
12 end
13 end
```
- Issue 3: A Ruby code snippet where an 'end' tag is not aligned with a 'group' tag. The code snippet shows:

```
17 group :jekyll_plugins do
18   gem 'jekyll-postfiles'
19 end
```

On the right side, there are filters for 'Category' (set to 'All Categories') and 'Engine' (set to 'All Engines').

Figure 37: CodeQuality service report for GitLab

5 Conclusion

5.1 Conclusion of possibilities for Gitlab as a Learning Management System

Website design

The goal of the visual design of the site was to make it clean, easy to navigate, and make it look in line with other NTNU affiliated websites. A simple white and blue color scheme was used to match the NTNU aesthetic.

The index page of the site contains a couple key elements that is presented front and center. The first thing a user sees is the announcements box. This is a conscious decision that serves to ensure that students will immediately know if there has been made a new announcement since the last time they visited the course's site. The other important part of the design is the navigation bar on the left side of the page layout. The role of the sidebar is to make the site easier to navigate and give the user a cursory overview of what information the site contains.

The links on the bar reflect the folder structure of the repository, allowing course coordinators to shape it to match whatever needs their course may have. The default folder structure that comes with the repository was decided upon based on a survey of students, existing sites referenced by Hjelmås, and the group's personal experience as students.

The page design is quite minimalist with a focus on functionality. In some places this serves the product, in others it is the result of limited time spent on the visual design. One known issue is that the navigation pages have no visual distinction between folders and files, which is a pretty significant user friendliness issue.

Through working with GitLab Pages for the duration of this project, the group finds that the visual website design is no more limited than when hosting a website in a more traditional manner. There's no reason why this solution couldn't be designed to fit any look a course coordinator would want. The Jekyll layouts and Sass files that make up the site's design are readily available for editing, and the default design of the developed system is already completely serviceable. For this aspect of developing a partial LMS alternative the group believes GitLab Pages is just

as suited as any other option.

Public info

How can information be made publicly available through using GitLab Pages?

Firstly, information is publicized on the World Wide Web which is available to all. The challenge of making course contents obtainable for students and teachers was solved with creating websites reachable on the World Wide Web. As a default, websites published as GitLab Pages are openly accessible for the public without restriction. Subsequently, the issue was minor in contrast to challenges introduced with transforming markdown files provided by teachers into websites.

Secondly, information regarded as public is given as input to the developed system and transformed with GitLab CI/CD into websites hosted as GitLab pages. Moreover, teachers trigger the GitLab CI/CD workflow whenever a committed change to the designated repository is sent. The workflow runs automated scripts, builds websites and reports on tests performed against the website and containers running the application. The primary challenge of dynamically creating static websites comes with document handling. For example, the pages are dynamically generated from teachers markdown files provided to the system. In addition, the web interface consists of pre-built HTML and CSS appearing as static websites on user's browser in contrast to dynamic pages which are built at the time of receiving the web request. With this in mind, several scripts including "deploy" in its naming was implemented in the GitLab CI/CD pipeline to overcome the obstacle of handling documents to become websites.

To summarize, GitLab enables developers to solve the challenges of publicizing public information as GitLab Pages and implement the automation necessary through GitLab CI/CD workflows. Group members have therefore concluded that GitLab Pages is a good alternative system for publicizing course content publicly.

Private info

How can a course coordinator handle access control to the three categories:

- a) Information available to the public

-
- b) information only for the participants in the course
 - c) information only for the course coordinator

The developed system applies access control to users within GitLab groups and projects. Which actions are permissible for a user is given by their role for a group or project. Furthermore, if multiple groups and projects are structured hierarchically roles are inherit from the top to bottom. For example, users part of the top level group inherits the same rights and permissions on the equivalent bottom level group. As a consequence, users needs to be given access to appropriate levels of group/project hierarchy for proper access control. It is noteworthy that defining user's roles in different part of the group/project hierarchy overrides or is added to the inherited permissions from a top-down perspective. For example, giving a user reading access on a top level group will affect all subsequent groups as well, and giving the same user writing access on a bottom level group makes reading and writing available for the user in that part of the hierarchy. Implementation of user access control in GitLab was made in reference to Mariusz Nowostawski setup available in appendix C.1. The proposed solution creates groups and projects for a course to support public, internal and private information. Moreover, information available to the public. Internal, information only for the participants in a given course. Private, information only for the course coordinator. In practice, course creation is done with scripted web requests sent to a specified GitLab instance and utilizes the user's own permissions.

All in all, user access control can be implemented with GitLab groups and projects where users inherits roles and permissions. Group members consider GitLab to be a suitable alternative system for solving challenges related to access control.

Announcements

The goal behind the announcement system is to provide teachers with a way of delivering important information to their students throughout the course. This is a very open prompt with many possible solutions. The developed system implements two different solutions side-by-side, with slightly different use cases.

The first implementation is closely inspired by the existing solution in BlackBoard Learn. The teacher creates a new markdown file in a specific announcement folder, the file is converted into

a static HTML site, and a link to the created site is added to the announcement box on the index page of the website. The second implementation is an automated variant that sends a message to a Teams channel whenever changes are made to the repository.

A big downside with the first implementation is that Git does not support sorting files by date. The scripts used for other markdown files all go by Git's default alphabetical sorting, which is not ideal for announcements, where the newest file should be on the top. Experimentation was done with using the files' creation times, but since the website is built inside a Docker instance the creation times reflect the creation of the instance, not the original source file. One possible solution to this problem is to demand users put the date in the file name of the announcement, or simply number the announcements. This is not implemented in the system developed for this thesis.

All in all the group believes announcements work well in GitLab. The date issue is the one part of the functionality that doesn't inherently do well in Git, but should there be a big demand for announcements along the line of traditional LMS, there are definitely ways to implement it. The group also believes the Teams integration is a good addition that could have real value. Multiple courses at NTNU already use Teams extensively, and this integration seems like a natural evolution of that.

Deliverables

Deliverables is the most complex functionality explored in this thesis. Implementing them properly necessitates significant architectural support that GitLab is not designed to offer. While there are ways to replicate the access control and actual file delivery aspects, there are also several other elements that are more problematic to implement.

The first of several issues comes with creating and enforcing deadlines for assignments. In our proposed solution to create a project per student for each assignment, enforcing a deadline would effectively take the form of updating the student's role within the project to no longer allow them to upload or delete files. Scheduling this kind of role change is not a functionality GitLab inherently supports. This means the teacher would either have to be online at that moment to make the change manually or there would have to be some kind of dedicated server that ran a scheduled script using the GitLab API. The group find neither of these solutions to be

satisfactory.

Another problematic element is the implementation of teacher feedback to student deliveries. This includes both a grade or score, and a way for teachers' to attach a written response to the deliverable. There is an obvious low-tech solution of having teachers create separate files in the assignment projects containing this information, but this makes it difficult for students to get an overview of their results. Traditional learning management systems like BlackBoard Learn will typically have an overview where a student can see all their completed and upcoming assignments on a single page, complete with grades and comments. The group found this very hard to replicate in GitLab, both on an architectural database level and on a presentation level, given the requirement of the website being entirely static.

This is where the strengths of systems built from the ground up to accommodate a certain use case shines. Without a dedicated self-defined database and certain key base functionalities some systems become very convoluted to implement. While deliverables are technically possible to support in GitLab, the group does not think it will be an optimal solution for every kind of assignment.

5.2 Evaluation

The group has work together several times before and have great knowledge on each others capabilities. As the group had worked on several projects before hand, working on this thesis did not pose a problem. At the start of the bachelor thesis the group made a contract dictation the amount of work the group have to use per week as a minimum.

For example, if the resulting product for the bachelor thesis did not meet the clients expectations. Could the group have put in more hours of work and development to create a better product.

5.2.1 Organizing

The group was able to fulfill their own project goals and reflects proper organizing throughout the project. For example, a vital part of utilizing the agile methodology is continuous improvement and development. The developed system and bachelor thesis was worked upon over

several months in time boxed periods. Each period had all group members partake in exercises to include all aspects into defining goals for thesis and developed system. Furthermore, the team was organized into three major roles and continuous improvement through reflective team exercises positively affected the group project as a whole. Scrum master was responsible for team coordination and ensured reaching the goals set. Developers realized through code the opportunities discovered by analysts. All in all, a symbioses between the organized roles group members attended was considered a success in retrospective of thesis.

5.2.2 Distribution of Workload

The group was in an extraordinary situation, as during the starting phase of the thesis work, one of the members were still abroad for their internship. It was not until March that all members were present in Norway and fully focusing on the thesis. In the meantime, the remaining three members were left with working short-staffed in the development process. Fortunately, considering the grand amount of research required to do in the starting phase, the member abroad was able to support the group through help with documentation and opportunity investigation, with the latter being assessment work to determine whether a product or service was worth including in the research.

Post-march, the workload was more even, but still within the same fields: The members who were central in the solution development were still focusing on solving the remaining challenges, with the remaining members focusing on documenting the development, as well as starting out with the thesis document. Both teams were working closely, to ensure that nothing is left undocumented. One major task of the documentation team was to also make models that could describe not only the solution workflow, but also its foundation, from an infrastructural perspective.

During the final weeks, all members were solely focused on writing on the thesis document. The workload was distributed based on what sections each member were supposed to write. As planned, there was minimal difference in how much workload each member received. However, it was noticeable that some members were quicker to write than others, and therefore decided to work on more sections throughout their working time.

5.2.3 Project as a form of work

With an entire semester designated for working with one project, and with high ambitions to score high, the group set out to work on a daily basis with expectations of weekends, from 11 AM until 4 PM, not mentioning overtime. In strong contrast to other standard projects provided in different courses, we were given a lot more writing support, at the expense of academical or product-related help as with other cases. Granted, we conducted a lot of interviews with professors with knowledge within our research field, but most of the insight provided was additional to already discovered findings.

Some obstacles were met with the cloud storage service GitLab, where during the month of April, the HTTPS license for the infrastructure at NTNU expired. This resulted in several CI jobs, namely the testing pipelines, in rendering as unavailable as they require an HTTPS connection. This slowed down our process on the testing, and the provisional development deadline had to be postponed by several weeks, as we had to wait for the infrastructure administrator to acquire a new license.

5.3 Reflection

5.3.1 Results

GitLab is surprisingly able to complete all the tasks given by the client, but it probably shouldn't. Announcements and deliveries where technically possible but complicated and lacked feature which is critical for a perfect product.

What worked well

GitLab pages in conjunction with Jekyll is well suited to display markdown documents as web pages and create a coordinated website. The setup is easy for use a teacher only adds their document to GitLab repository and the site is generated. For the student the website works like any other.

Private information is not ideal in the current solution because it is depended on a role base segregation where a student reads from markdown documents in a repository closed off to only course affiliated. Being able to move away from reading markdown documents directly from

GitLab is part of the reason the client wanted to explore the possibilities with GitLab pages. However, GitLab pages is by default configured (for the GitLab instance, configuring what a GitLab user can configure in their project) to public only, but this is able to change and thus give the private information the ability to be created as a website.

What did not work well

Deliveries without the use of a database and authentication, is entirely trust based. The current solution would clone students projects to the course coordinator's computer, to know who a delivery belong to a student have to include some form of authentication in the delivery. In addition, a teacher do not possess a centralized sheet of which students passed which tasks, and have no automatic way of knowing if a student qualify for the courses exam.

Announcements are also possible but without persistent data it becomes impossible to create a system showcasing priority among the announcements. Because the solution cannot save data, like a date variable, between instances announcements will have an issue if a course's exam date changes twice. Then two announcements are sent and students will not know which announcement is applicable as they cannot be dated and sorted.

5.3.2 Discoveries

Though the projects duration the group made a series of discoveries, both from a developing point of view and also as the customer, when choosing the right product.

Importance of data persistence

Announcements had an issue with data persistence, where displaying the creation date of the announcement where not possible with the current setup. Additionally Jekyll sort files by name and when there is no other value, a system for announcements to be sorted by date (where the newest announcement is at the top) is impossible.

Mattermost v MS Teams

Mattermost is an online chatting service alternative to MS Teams. NTNU has a license for and uses Mattermost under the same domain as the Gitlab instance the group used for the project (stud.idi.ntnu). However, Mattermost is not widely used and request (similar to MS Teams webhook requests) to the same domain (local requests) is not allowed on this domain. Therefore

the group went for MS Teams as its third party announcement service. Besides, Teams is more widely used by teachers, and given the extended possibilities with webhook implementation the solution opened up for customisation of how much and how the information is displayed to the students.

Pandoc v Jekyll

In the beginning the group experimented with plain HTML and scripts using pandoc to convert markdown to HTML pages. Pandoc is a service which converts files from one markup format into another [60], however this method quickly created a need for a lot of script to aid the website. As a way to alleviate the need for script the group switched to Jekyll.

5.4 Critique of task

The group members consider the thesis task to be of an appropriate difficulty and scope for four students of Digital Infrastructure and Cybersecurity. For example, the research conducted with regard to the thesis was positively affected by group members' competence acquired from their bachelor degree. Furthermore, the developed system is hosted on digital infrastructure and involves several cybersecurity principles such as user access control and vulnerability scanning.

The task itself is rather open-ended. The group members were allowed to research freely how to solve the challenges posed in the task description. One could therefore argue that the lack of descriptive guidelines and requirements positively affected the thesis, but there were also a couple of downsides. For example, the less specific the task, the harder it is to validate the success of the finished product. Moreover, the requirements were solely an idea of group members to include in the thesis and were developed with the client thereafter. After all, it is imperative in a research setting for subsequent projects to be able to produce similar or contradicting results to affirm the research questions posed in the thesis. All in all, the group believes a good job was done to explore the research questions, and enjoyed the freedom that resulted from the open-ended nature of the task.

5.5 Future Considerations

Future iterations of the developed system should incorporate a study to identify the students response to current website layout and design. Depending on the results, websites could require an aesthetic rework.

An obstacle that could become a future consideration is related to routine based automation. For example, to fully support the functionality of delivering tasks in GitLab one would need to change the roles of users on designated times. Furthermore, students could have the role of developer and be able to upload and change their deliveries until the deadline. Afterwards, their roles would only include reading access. An improvement of the current iteration could therefore incorporate automated and routine based scripting and lay the foundation for supporting the functionality of delivering tasks.

Announcements on the index page are at the moment listed in alphabetical order. The group could not find a simple way of sorting the announcements based on their creation dates as this data is not readily available. At the same time there is almost definitely some way of working around this by either adding the date to the file name or the file's contents on creation, or any other such way of forcefully adding the data to the file. Should the developed system see actual use, this is probably high on the list of revisions that could considerably improve the site's usability.

During the span of the project, a frequent question occurred about whether it is feasible to continue developing the solution after the thesis, and potentially make it a full-fledged product. Despite the support from the academical community of NTNU, and a head start with a working solution, the group has decided to not continue working on the solution. Reason behind this decision is primarily the lack of interest in developing such a solution, as the group members have passion for other fields within information technology, and would additionally lack the time to continue working on the project as other opportunities arise. Everyone in the group have received a job offer for the following months, and would rather pursue that than continue working on improved solutions to BlackBoard. Nevertheless, we encourage others to follow our steps and improve the existing solution into a better product.

References

- [1] Department of Marine Technology NTNU. *IMT Software Wiki - LaTeX*. URL: <https://www.ntnu.no/wiki/display/imtsoftware/LaTeX> (visited on 15th Sept. 2020).
- [2] Wikipedia contributors. *Educational technology*. URL: https://en.wikipedia.org/w/index.php?title=Educational_technology&oldid=1088592261 (visited on 19th May 2022).
- [3] NTNU. *Bachelor of Science in Digital Infrastructure and Cyber Security*. URL: <https://www.ntnu.edu/studies/bdigsec> (visited on 8th May 2022).
- [4] GitLab. *GitLab CI/CD*. URL: <https://docs.gitlab.com/ee/ci/> (visited on 3rd May 2022).
- [5] Gitlab. *Gitlab Pages*. URL: <https://docs.gitlab.com/ee/user/project/pages/> (visited on 9th May 2022).
- [6] Wikipedia contributors. *Jekyll (software)*. URL: [https://en.wikipedia.org/w/index.php?title=Jekyll_\(software\)&oldid=1073370718](https://en.wikipedia.org/w/index.php?title=Jekyll_(software)&oldid=1073370718) (visited on 19th May 2022).
- [7] Shopify. *Liquid reference*. URL: <https://shopify.dev/api/liquid> (visited on 5th May 2022).
- [8] Wikipedia contributors. *cgroups*. URL: <https://en.wikipedia.org/w/index.php?title=Cgroups&oldid=1076635510> (visited on 8th May 2022).
- [9] Wikipedia contributors. *Central processing unit*. URL: https://en.wikipedia.org/w/index.php?title=Central_processing_unit&oldid=1088489284 (visited on 8th May 2022).
- [10] Docker. *The underlying technology*. URL: <https://docs.docker.com/get-started/overview/#the-underlying-technology> (visited on 8th May 2022).
- [11] Wikipedia contributors. *Linux kernel*. URL: https://en.wikipedia.org/w/index.php?title=Linux_kernel&oldid=1088409007 (visited on 8th May 2022).
- [12] Docker. *What does Docker technology add to just plain LXC?* URL: <https://docs.docker.com/engine/faq/#what-does-docker-technology-add-to-just-plain-lxc> (visited on 8th May 2022).
- [13] Vladimir Baranek. URL: <https://www.linkedin.com/in/vladimir-baranek-15a6a54/> (visited on 8th May 2022).

-
- [14] Vladimir Baranek. In: (). URL: <https://www.linkedin.com/pulse/top-benefits-containerization-vladimir-baranek/> (visited on 8th May 2022).
- [15] Nicole Hemmer. URL: <https://linfordco.com/blog/containerization-security/> (visited on 8th May 2022).
- [16] Wes Felter et al. *An Updated Performance Comparison of Virtual Machines and Linux Containers*. IEEE, 2015.
- [17] Khan Academy. *The Scientific Method*. URL: <https://www.khanacademy.org/science/biology/intro-to-biology/science-of-biology/a/the-science-of-biology> (visited on 19th May 2022).
- [18] Paul Clements et al. *Documenting Software Architectures: Views and Beyond, Second Edition*. Addison-Wesley, 2010.
- [19] Carnegie Mellon University. *Software Architecture*. URL: <https://www.sei.cmu.edu/our-work/software-architecture/> (visited on 10th May 2022).
- [20] Bob Dignen. *Five reasons why feedback may be the most important skill*. URL: <https://www.cambridge.org/elt/blog/2014/03/17/five-reasons-feedback-may-important-skill/> (visited on 11th May 2022).
- [21] Gitlab. *Permissions and roles*. URL: <https://docs.gitlab.com/ee/user/permissions.html> (visited on 13th May 2022).
- [22] Pa11y. *Pa11y CI*. URL: <https://github.com/pa11y/pa11y-ci> (visited on 13th May 2022).
- [23] Snyk. *SNYK Security Intelligence*. URL: <https://snyk.io/snyk-intelligence-security/> (visited on 13th May 2022).
- [24] Frode Haug. *IDATG2102 - Algoritmiske metoder*. URL: <https://folk.ntnu.no/frh/algmet/> (visited on 19th May 2022).
- [25] w3schools. *HTML <footer> Tag*. URL: https://www.w3schools.com/tags/tag_footer.asp (visited on 6th May 2022).
- [26] Brian Rashid. *5 Essential Reasons You Should Be Using A Responsive Website Design Now*. URL: <https://www.forbes.com/sites/brianrashid/2017/06/13/5-essential-reasons-and-benefits-why-you-should-be-using-a-responsive-website-design-now/?sh=b4e8c417c91e> (visited on 4th May 2022).

-
- [27] Docker. *Docker Hub Quickstart*. URL: <https://docs.docker.com/docker-hub/> (visited on 9th May 2022).
- [28] Robert Gibb. *What is Runtime?* URL: https://en.wikipedia.org/w/index.php?title=Runtime_system&oldid=1082900964 (visited on 13th May 2022).
- [29] O'Reilly. *Understanding upstream and downstream jobs*. URL: <https://www.oreilly.com/library/view/jenkins-2x-continuous/9781788297943/af3de8e1-f3e9-4089-8e03-609cef1c0444.xhtml> (visited on 10th May 2022).
- [30] GitLab. *Set up your development environment*. URL: <https://docs.gitlab.com/omnibus/development/setup.html> (visited on 14th May 2022).
- [31] Fortinet. *Endpoint Detection and Response (EDR) Defined*. [Online; Accessed 02-May-2022]. URL: <https://www.fortinet.com/resources/cyberglossary/what-is-edr>.
- [32] Cisco. *Cisco Secure Endpoint (formerly AMP for Endpoints) At-a-Glance*. [Online; Accessed 02-May-2022]. URL: <https://www.cisco.com/c/en/us/solutions/collateral/enterprise-networks/advanced-malware-protection/at-a-glance-c45-731874.html>.
- [33] Gitlab. *Password Storage*. URL: https://docs.gitlab.com/ee/security/password_storage.html (visited on 22nd Apr. 2022).
- [34] Michael Gegick and Sean Barnum. *Separation of Privilege*. URL: <https://www.cisa.gov/uscert/bsi/articles/knowledge/principles/separation-of-privilege> (visited on 2nd May 2022).
- [35] Microsoft. *Create an Incoming Webhook*. URL: <https://docs.microsoft.com/en-us/microsoftteams/platform/webhooks-and-connectors/how-to/add-incoming-webhook> (visited on 18th May 2022).
- [36] Gitlab Docs. *Shared runners*. URL: https://docs.gitlab.com/ee/ci/runners/runners_scope.html (visited on 19th May 2022).
- [37] GitLab. *Securit for self-managed Runners*. URL: <https://docs.gitlab.com/runner/security/> (visited on 8th May 2022).
- [38] Wikipedia contributors. *Variable (computer science)*. URL: [https://en.wikipedia.org/w/index.php?title=Variable_\(computer_science\)&oldid=1086223546](https://en.wikipedia.org/w/index.php?title=Variable_(computer_science)&oldid=1086223546) (visited on 10th May 2022).
-

-
- [39] Gitlab. *Multi-project pipelines*. URL: https://docs.gitlab.com/ee/ci/pipelines/multi_project_pipelines.html (visited on 10th May 2022).
- [40] GitLab. *Permissions and Roles*. URL: <https://gitlab.stud.idi.ntnu.no/help/user/permissions> (visited on 10th May 2022).
- [41] auth0. *Use Access Tokens*. URL: <https://auth0.com/docs/secure/tokens/access-tokens/use-access-tokens> (visited on 10th May 2022).
- [42] Gitlab. *Gitlab Pages access control*. URL: https://docs.gitlab.com/ee/user/project/pages/pages_access_control.html (visited on 9th May 2022).
- [43] Gitlab. *Gitlab Pages administration*. URL: <https://docs.gitlab.com/ee/administration/pages/index.html#access-control> (visited on 9th May 2022).
- [44] HackerOne. *Ruby Bug Bounty program*. URL: <https://hackerone.com/ruby?type=team> (visited on 18th Apr. 2022).
- [45] et al. Iván Arce. *Avoiding the Top 10 Software Security Design Flaws*. URL: <https://ieeecs-media.computer.org/media/technical-activities/CYBSI/docs/Top-10-Flaws.pdf> (visited on 10th May 2022).
- [46] NVD. *CVE-2018-17567 Detail*. URL: <https://nvd.nist.gov/vuln/detail/CVE-2018-17567> (visited on 18th Apr. 2022).
- [47] GitLab. *Static Vs Dynamic Websites*. URL: <https://about.gitlab.com/blog/2016/06/03/ssg-overview-gitlab-pages-part-1-dynamic-x-static/> (visited on 16th May 2022).
- [48] CAPEC. *CAPEC-66: SQL Injection*. URL: <https://capec.mitre.org/data/definitions/66.html> (visited on 18th May 2022).
- [49] NTNU. *Employees*. URL: <https://www.ntnu.edu/employees/henrik.johnsen> (visited on 9th May 2022).
- [50] Xing Gao et al. ‘Houdini’s Escape: Breaking the Resource Rein of Linux Control Groups’. In: (). URL: <https://www.cs.memphis.edu/~xgao1/paper/ccs19.pdf#page=13&zoom=100,428,380> (visited on 9th May 2022).
- [51] Snyk. *How do you pronounce Snyk?* URL: <https://support.snyk.io/hc/en-us/articles/360000890358-How-do-you-pronounce-Snyk-> (visited on 9th May 2022).
- [52] Snyk. *Partnerships for best in class security*. URL: <https://snyk.io/partners/> (visited on 9th May 2022).
-

-
- [53] Snyk. *Snyk Security Intelligence Integrates into the new, enhanced Amazon Inspector*. URL: <https:// snyk .io / news / snyk - security - intelligence - integrates - into - the - new - enhanced - amazon - inspector /> (visited on 9th May 2022).
- [54] Bundler. *Docs*. URL: <https://bundler.io/man/gemfile.5.html> (visited on 9th May 2022).
- [55] Docker. *Docker Official Images*. URL: https://docs.docker.com/docker-hub/official_images/ (visited on 9th May 2022).
- [56] the Docker Community. *ruby*. URL: https://hub.docker.com/_/ruby (visited on 9th May 2022).
- [57] w3. *WCAG 2 Overview*. URL: <https://www.w3.org/WAI/standards-guidelines/wcag/#intro> (visited on 10th May 2022).
- [58] Gitlab. *Accessibility testing*. URL: https://docs.gitlab.com/ee/user/project/merge_requests/accessibility_testing.html (visited on 10th May 2022).
- [59] GitLab. *GitLab AutoDevOps*. URL: <https://docs.gitlab.com/ee/topics/autodevops/index.html> (visited on 23rd Apr. 2022).
- [60] Pandoc. *About pandoc*. URL: <https://pandoc.org/> (visited on 18th May 2022).
- [61] Citrix Systems Inc. *What is containerization?* URL: <https://www.citrix.com/no-no/solutions/app-delivery-and-security/what-is-containerization.html> (visited on 6th May 2022).
- [62] Wikipedia contributors. *DevOps*. URL: <https://en.wikipedia.org/w/index.php?title=DevOps&oldid=1087274706> (visited on 10th May 2022).
- [63] Wikipedia contributors. *Docker (Software)*. URL: [https://en.wikipedia.org/w/index.php?title=Docker_\(software\)&oldid=1086277493](https://en.wikipedia.org/w/index.php?title=Docker_(software)&oldid=1086277493) (visited on 6th May 2022).
- [64] Gitlab Docs. *GitLab Runner*. URL: <https://docs.gitlab.com/runner/> (visited on 19th May 2022).
- [65] Defuse Security. *Salted Password Hashing - Doing it Right*. URL: <https://crackstation.net/hashing-security.htm> (visited on 2nd May 2022).
- [66] Programming Think. *Cryptography series: detailed explanation of bcrypt encryption algorithm*. URL: <https://programmer.ink/think/cryptography-series-detailed-explanation-of-bcrypt-encryption-algorithm.html> (visited on 2nd May 2022).

-
- [67] Wikipedia contributors. *Jira (software)*. URL: [https://en.wikipedia.org/w/index.php?title=Jira_\(software\)&oldid=1086177680](https://en.wikipedia.org/w/index.php?title=Jira_(software)&oldid=1086177680) (visited on 19th May 2022).
- [68] Jekyll. *Jekyll Layouts*. URL: <https://jekyllrb.com/docs/step-by-step/04-layouts/> (visited on 5th May 2022).
- [69] markdownguide.org. *Getting Started*. URL: <https://www.markdownguide.org/getting-started/> (visited on 8th May 2022).
- [70] Microsoft Support. *What is Microsoft Teams?* URL: <https://support.microsoft.com/en-us/topic/what-is-microsoft-teams-3de4d369-0167-8def-b93b-0eb5286d7a29> (visited on 19th May 2022).
- [71] Microsoft Docs. *Cards*. URL: <https://docs.microsoft.com/en-us/microsoftteams/platform/task-modules-and-cards/what-are-cards> (visited on 19th May 2022).
- [72] Steve Ovens. *The 7 most used Linux namespaces*. URL: <https://www.redhat.com/sysadmin/7-linux-namespaces> (visited on 6th May 2022).
- [73] NPM. *About NPM*. URL: <https://docs.npmjs.com/about-npm> (visited on 9th May 2022).
- [74] Node.js®. *Home*. URL: <https://nodejs.org/en/> (visited on 9th May 2022).
- [75] Wikipedia contributor. *OAuth*. URL: <https://en.wikipedia.org/w/index.php?title=OAuth&oldid=1088647506> (visited on 9th May 2022).
- [76] Wikipedia contributors. *Operating system*. URL: https://en.wikipedia.org/w/index.php?title=Operating_system&oldid=1088674736 (visited on 6th May 2022).
- [77] MITRE. *Phishing, Technique T1566 - Enterprise | MITRE ATT&CCK®*. URL: <https://attack.mitre.org/techniques/T1566/> (visited on 18th May 2022).
- [78] Remzi H. Arpaci-Dusseau and Andrea C. Arpaci-Dusseau. *What is containerization?* URL: <https://pages.cs.wisc.edu/~remzi/OSTEP/cpu-intro.pdf> (visited on 6th May 2022).
- [79] Microsoft. *What is SaaS?* URL: <https://azure.microsoft.com/en-us/overview/what-is-saas/> (visited on 16th May 2022).
- [80] Sass ©. *Documentation*. URL: <https://sass-lang.com/documentation> (visited on 4th May 2022).

-
- [81] Mozilla Web Docs. *Element*. URL: <https://developer.mozilla.org/en-US/docs/Glossary/Element> (visited on 4th May 2022).
- [82] Vijayakumar Nanjappan et al. *Big Data Analytics for Sensor-Network Collected Intelligence*. Academic Press, 2017, pp. 3–20.

A Definition of Done

A.1 Functionality to create website from markdown

Functionality to create website from markdown

1. Define functionality

1.1 Purpose – why do we need this functionality

A core functionality of learning management systems is access to information. Public information should contain all necessary contents for stakeholders in each subject.

Information to be regarded as public is defined as the following:

- Announcements
- Curriculum
- Course coordinator's contact information

1.2 Intended Audience

Students and employees enrolled in a specified subject.

1.3 Intended Use

Members of a subject are given access to all information defined to be publicly available.

Furthermore, all learning material, schedule and subject description that is marked or structurally designed to be public is intended to be used freely by the students/employees.

1.4 Assumptions and Dependencies

Schedules are individual for a given subject. Furthermore, a schedule can be textually represented with static information. As opposed to having an overall dynamic schedule interlinking multiple subjects on a student/employee.

Access to the public folder depends on a basic user access control for students/employees in the given subject. For example, NTNU's Gitlab is foremost available with connection to the internal network which is only available for enrolled students and employees.

2. System Features and Requirements

2.1 Functional Requirements

1. Convert markdown files to html pages and host these as a website.
2. Information regarded as "Public" should be available for all users, without any restrictions.
3. Course coordinator can freely remove or add public information as needed.
4. Gitlab pages portraying public information should be designed according to WCAG 2.0 and Universal design of ICT at NTNU guidelines on web content availability at AA level or above, international standard 40500.2012 (ISO/IEC).
5. Webpages should have a structure typical for an academical site, in line with other NTNU affiliated websites.

2.2 Non-functional Requirements

Performance

- Instantaneous or loading screen.

Safety

- Should not allow sharing of private information.

Security

- Implementation of security as followed by NTNU standards

Quality

- Solution is to follow design principles of Don Norman to ensure satisfaction from both parties of the product.

IxD

- Don Norman's list of principles (for websites)
- Point 4 of functional requirements

An important notice is that some of these requirements are dependent on the infrastructure provider (Performance).

A.2 Functionality to support access control

Functionality to support access control

1. Define functionality

1.1 Purpose

Provide easy access to closed information (solution proposals, non-public curriculum, copyrighted material, etc.).

For example, a teacher can separate private information from public and to only disclose that data to selective groups.

1.2 Intended Audience

Repository owner/teachers

1.3 Intended Use

Teachers can upload documents to only be available to designated persons/groups.

Students can only access private information that is disclosed to them or published as public.

1.4 Assumptions and Dependencies

IMPORTANT! We assume Gitlab does not support role-based systems. In such scenarios an external access-control system might be necessary or developing new software to control access for Gitlab repositories. For example:

- A ruby plugin that retrieves data from an NTNU API/FEIDE...
 - Gitlab .stud.idi.no... bruker feide
- Javascript with access to own database or external – in essence cookies on browsers after authentication
- A private repository that contains all information, only repository owner (usually teacher) can access the information. Specify in Jekyll to only make websites and related documents from a specific folder in directory structure inside repository.
- Multiple repo solution. One private repo on NTNU's network, that requires being on location or VPN to access. The GitLab page built from this repo will only be available to students or other persons affiliated with NTNU. Deploying to this repo will trigger a pipeline that also builds relevant files to a completely public repo on another network, which builds a page available to the public.
- Gitlab pages can limit the accessibility of the website to members of the git repository which the website is created from. However, to enable this feature a git administrator needs to enable a user to limit access of website to members only.

Pages access control is disabled by default. To enable it:

1. Enable it in `/etc/gitlab/gitlab.rb`:

```
gitlab_pages['access_control'] = true
```

2. System Features and Requirements

2.1 Functional Requirements

The goal is to have three degrees of access:

1. Public – available to everyone
2. Private – available to enrolled students
3. Super private – available to Course responsible

2.2 Non-functional Requirements

If Gitlab does not provide any methods of access control, development of new software would require substantial research and frequent updates for the following subjects:

- Performance
- Safety
 - Private information is required to keep its integrity within the repository, without being available any other place than the one it was designed for.
- Security
 - The security aspect of Private information is aimed at keeping the objects secure from issues with confidentiality, integrity and availability. Respectively, the customer's files should be inaccessible from outside of the environment, should be untamperable beyond repair and not subject to being lost due to technical issues.
- Quality

Preferably, we can use an external API and relate non-functional requirements to their solution.

3. Existing Solution in blackboard

Feide is provided by Sikt - the knowledge sector's service provider. The service enables secure login and data sharing in education and research. They collaborate with the Directorate of Education on the administration of the service. Using Feide's existing solution would be highly preferable for implementing private information if possible.

A.3 Functionality to publish announcements

Functionality to publish announcements

1. Define functionality

1.1 Purpose

The purpose of the announcement functionality is to give educators a way of publishing information concerning changes or other important information that appears underway in the course. They should encompass important information that should be hard to miss.

1.2 Intended Audience

Published announcements are directed towards students enrolled in a course.

1.3 Intended Use

The intention is for teachers to push out a small text file containing important or time sensitive information that should be seen by every student enrolled in the course as fast as possible.

1.4 Assumptions and Dependencies

We assume a teacher sends, hence know of the announcements.

Announcements should only relate to a subject and only be transmitted the enrolled students of that subject.

2. System Features and Requirements

2.1 Functional Requirements

For example:

1. Students should be able to track/view changelog of changes in course
2. Newly created announcements should be easily visible to students, preferably on the index page
3. Announcements should be pushed to a dedicated announcement folder in the Git repo
 - a. Pushes to this folder should be detected automatically
4. Functionality should be included to send mail to all students enrolled in course

3. Existing Solution in blackboard

The existing solution in Blackboard is very similar to the one we plan to implement ourselves as we find it to be a pretty good solution.

A.4 Functionality to handle assignments

Functionality to handle assignments

1. Define functionality

1.1 Purpose

Assignments are tasks given by a course responsible, for example teachers. Students are expected to deliver their answer to a given assignments and receiving feedback afterwards.

1.2 Intended Audience

Students, student assistants and teachers. Student will be able to submit their work and student assistants and teachers can view the submitted work.

1.3 Intended Use

Students will be able to submit their work and student assistants and teachers can view the submitted work. A teacher should be able to give feedback to the student. Teacher should control that alle groups are correct, for instance no group exists with more members then permitted.

1.4 Assumptions and Dependencies

Students should only be able to view their separate delivered documents related to an assignment. Despite this, during peer review the student pair should be able to read each others entries.

2. System Features and Requirements

2.1 Functional Requirements

For example:

1. All students enrolled in a subject must have access to a project/task folder with a gitlab project assigned to them.
2. A teacher can automatically lock students project for further development as a deadline.
3. A teacher can give feedback to student project.
4. A teacher should be able to control the amount of members in task-projects to ensure no group is illegal.
5. Automated solutions should be given to teachers for conducting peer-reviews.
6. Teachers can withdraw information of which students have delivered and fulfilled the requirements for an assignment.

2.2 Non-functional Requirements

Performance: A student must be able to submit their work. A script for deadlines must be precis to the deadline.

Safety

Security: A student should have access to their work after it is submitted but not be able to submit further changes. A teacher should control members of projects and ensure there are no illegal groups.

Quality

B Survey

Learning Management System (LMS)

23
Responses

04:46
Average time to complete

Active
Status

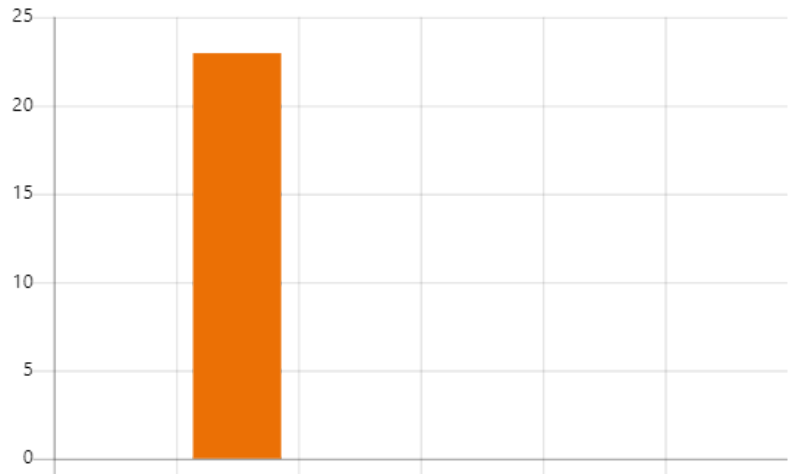
1. What gender are you?

● Woman	6
● Man	17
● Prefer not to say	0



2. What is your age?

● Under 18	0
● 18-24	23
● 25-34	0
● 35-44	0
● 45-55	0
● 56+	0



3. What LMS do you mostly use?

● Blackboard	8
● Itslearning	2
● Fronter	0
● Canvas	10
● Other	3



4. How has your overall experience been with your LMS?

★ = Terrible

★★★★★★★★★★★★ = Excellent

23

Responses



6.17 Average Rating

5. Rank the functionalities in order of what you use your LMS for

Highest

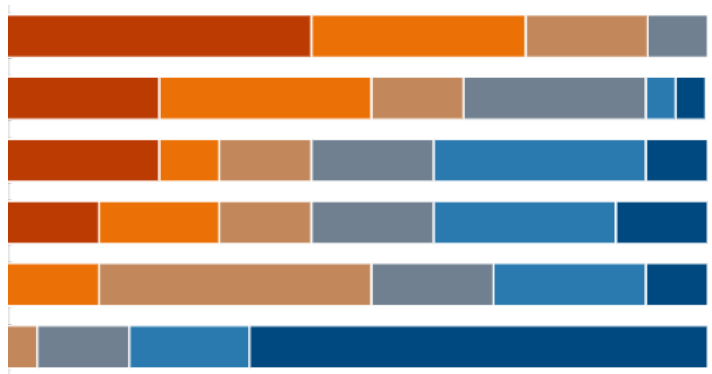


Lowest

Rank Options

- 1 Finding tasks for a course
- 2 Handing in tasks
- 3 Looking at calendar for lectures
- 4 Online lectures
- 5 Reading course material (curri...
- 6 Collaborate with students, wor...

First choice ■ ■ ■ ■ ■ ■ Last choice



6. Are there any functionalities you would like to be implemented in your LMS? Or improvements of existing functionalities?

16

Responses

Latest Responses

""

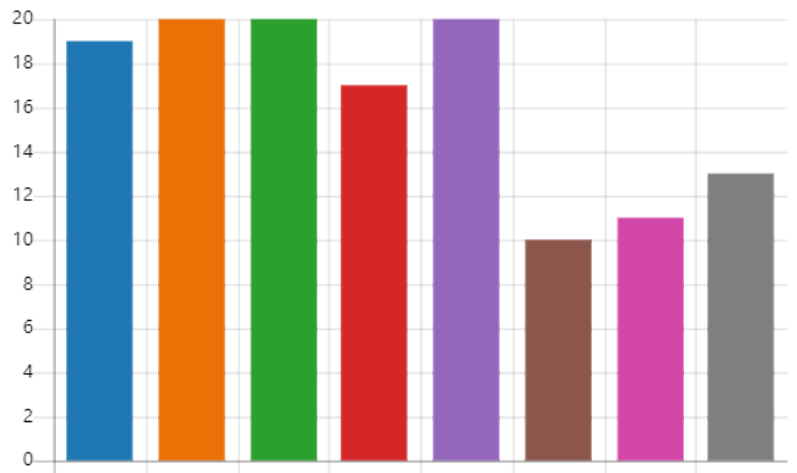
"Gjøre alt så enkelt så mulig å orientere seg i. Typ sånn som onenote e...

4 respondents (25%) answered **app** for this question.



7. Which links would you want a course home page to contain?

● Lectures/Arrangements	19
● Assignments	20
● Curriculum	20
● Course Information	17
● Resources	20
● Learning assistans	10
● Student evaluation of course	11
● Suggested solutions for tasks	13



ID	Start time	Completion time	What LMS do you mostly use?	How has your overall experience been with your LMS? ★ = Terrible ★★★★★★★★ = Highest	Rank the functionalities in order of what you use your LMS for	Are there any functionalities you would like to be implemented in your LMS? Or improvements of existing functionalities?	Which links would you want a course home page to contain?	Are there any other relevant links not mentioned above a course home page should contain?2	What gender are you?	What is your age?
8	4/5/22 11:34:29	4/5/22 11:34:39	Fronter		course;Looking at calendar for lectures;Handin g in tasks;Reading saAS		Student evaluation of course;Course Information;Curriculum;	ASDASD	Woman	45+
9	4/5/22 15:39:24	4/5/22 15:40:01	Blackboard		for lectures;Finding tasks for a course;Reading course material as		Course Information;Lectur es/Arrangements;A ssignments;	asdsa	Man	18-24
10	4/16/22 16:30:20	4/16/22 16:32:35	Canvas		for lectures;Finding tasks for a course;Reading course material		Information;Curric ulum;Assignments; Lectures/Arrangem ents;Resources;Stu		Man	18-24
11	4/16/22 17:02:01	4/16/22 17:04:21	Canvas		course;Handing in tasks;Collaborate with students, 7 working on		ents;Assignments; Curriculum;Course Information;Resour ces;Suggested		Man	18-24
12	4/16/22 17:01:44	4/16/22 17:06:47	Blackboard		course;Handing in tasks;Looking at calendar for lectures;Online	Mer stabil app. Slitsomt med en app som ikke funker annenhver dag.	ents;Assignments; Curriculum;Course Information;Resour ces;Learning	Karakterskala, diskusjonsforu m	Man	18-24
13	4/17/22 15:26:57	4/17/22 15:29:04	Blackboard		course;Reading course material (curriculum);Onlin e lectures;Handing tasks;Finding tasks for a course;Online lectures;Collaborat e with students,	Mindre utlogging. Blir kastet ut ganske ofte Ha BB kurs med foreleseme plis, alle bruker det forskjellig og gjør alt feil.	ents;Assignments; Curriculum;Course Information;Resour ces;Learning		Man	18-24
14	4/18/22 11:29:40	4/18/22 11:31:47	Blackboard		tasks;Finding tasks for a course;Online lectures;Reading course material	Enklere og mer oversiktlig design. Enklere å laste ned filer.	ng assistants;Course Information;Resour ces;Assignments;S	Karakterstatistik	Man	18-24
15	4/18/22 11:29:40	4/18/22 11:34:10	Blackboard		5 course material		uggested solutions		Man	18-24

16	4/18/22 11:36:11	4/18/22 11:47:15	Blackboard	lectures;Handing in tasks;Reading course material	Mer oversiktlig/bedre design på chat forum	ents;Suggested solutions for tasks;Course Information;Curriculum;Lectures/Arrangements;Suggested solutions for tasks;Resources;		Woman	18-24
17	4/18/22 11:47:33	4/18/22 11:51:55	Canvas	8 (curriculum);Finding for lectures;Handing in tasks;Finding tasks for a course;Online	LMS, but i think the generelt er ganske rotete. Er ikke så viktig med uendelige	res/Arrangements; announcement	Kalender	Man	18-24
18	4/18/22 11:29:13	4/18/22 11:53:03	Blackboard	8 (curriculum);Online course;Looking at calendar	phone app has a bad interface as it is hard to navigate in	Curriculum;Course Information;Resources;Learning	link which has a overview of recent activity	Man	18-24
19	4/18/22 11:53:15	4/18/22 12:02:50	Canvas	4 g in tasks;Online course;Looking at calendar	enklere å få kalenderen sin opp med alle typer arrangementer	Lectures/Arrangements;Assignments;Curriculum;	inspo fra sosiale medier, men at det er dedikert	Man	18-24
20	4/18/22 12:35:08	4/18/22 12:36:48	Itslearning	8 for lectures;Handing in tasks;Reading course material (curriculum);Finding tasks for a	Lettere oversikt i ressurser	Curriculum;Resources;Suggested solutions for tasks;		Man	18-24
21	4/18/22 13:54:19	4/18/22 13:57:07	Teams	7 g tasks for a	courses one have taken in canvas even after completing the course	Information;Curriculum;Lectures/Arrangements;Resources;		Woman	18-24
22	4/18/22 13:56:09	4/18/22 14:04:40	Canvas	8 course;Reading for lectures;Online lectures;Handing in tasks;Finding tasks for a	en app som fungerer (vi hadde app for som var omtrent ubrukelig, men den ble fjernet	ents;Assignments; Curriculum;Course Information;Resources;Learning		Man	18-24
23	4/18/22 14:57:39	4/18/22 15:05:11	Canvas	6 course;Handing in tasks;Reading course material		Assignments;Lectures/Arrangements; Resources;		Woman	18-24
24	4/18/22 15:33:46	4/18/22 15:35:11	Canvas	6 (curriculum);Looking				Man	18-24

25	4/18/22 17:45:51	4/18/22 17:46:52	Canvas	course;Handing in tasks;Reading course material	Bedre mappestifunksjonalitet	ents;Assignments; Curriculum;Course Information;Resources;Suggested	Man	18-24
26	4/18/22 20:17:02	4/18/22 20:18:12	team	lectures;Finding tasks for a course;Reading course material		Student evaluation of course;	Woman	18-24
27	4/18/22 22:48:40	4/18/22 22:50:49	Itslearning	course;Online lectures;Handing in tasks;Reading course material	Brukervennlighet	of course;Suggested solutions for tasks;Curriculum;A	Man	18-24
28	4/19/22 10:13:16	4/19/22 10:21:47	Blackboard	course;Looking at calendar for lectures;Reading course material	Blackboard appen var flinkere til å gi notifications på announcements i	ents;Assignments; Curriculum;Course Information;Resources;Learning og hvordan de	Man	18-24
29	4/19/22 11:20:54	4/19/22 11:23:03	Canvas	g course material (curriculum);Finding tasks for a		ents;Assignments; Curriculum;Resources;Course Information;	Woman	18-24
30	4/20/22 13:29:13	4/20/22 13:30:39	Feide	tasks;Finding tasks for a course;Reading course material	Gjøre alt så enkelt så mulig å orientere seg i. Typ sånn som onenote er for skriving	Curriculum;Assignments;Resources;	Man	18-24
31	4/22/22 22:36:16	4/22/22 22:39:46	Canvas	lectures;Finding tasks for a course;Looking at calendar		ents;Resources;Course Information;Curriculum;Assignments;	Woman	18-24

C Interviews

C.1 Mariusz Nowostawski

Asked for privilege to quote in bachelor thesis, we will use to support our report – **ALLOWED**

Asked to use name - **ALLOWED**

Interview Mariusz Nowostawski

1. purpose of interview – high level about project

Group members presented thesis

2. Discussion of Mariusz Nowostawski's access control design in Gitlab:

First iteration of Mariusz ACL setup worked directly with gitlab api:

- Ultimately had major issues
- Gitlab updates their api often which would require frequent updates of the scripts involved

second iteration:

- Directly on gitlab instance and uses gitlab console
- Ruby based, which is considered stable

Gitlab functionalities can be extended with ruby. Sideeffects: always work, as it is directly with gitlab and does not update. Can argue that a downside is scripts are ran as sudo users.

Recommendations for development of system for thesis?

Mariusz recommends graphql or gitlab api, but unsure if it actions to modify documents in gitlab works. Gitlab actions for reading will definitely work.

Group members received access to Mariusz repository containing the Ruby scripts in their ACL solution.

The current ACL solution does not automate the process of retrieving students, why not?

Deliberate decision to not automate. As it indirectly "checks" which students are active in a course. They have instead a process on how to do it. For example a wiki that explains the workflow, and usually after the first semester students understands how it works.

In a prior iteration administrators on Gitlab at NTNU had automation for retrieving students in a course. two side effects, created groups had students that was not active. for example automatic peer review becomes problematic when some students are inactive.

Discussion on the primary differences between system for thesis and Mariusz ACL setup

Mariusz use wiki instead of gitlab pages. right hand side of menu, wiki for content itself like videos. Git repo for code examples, and issue tracker.

Mariusz does use in some subjects, for example cloud computing, use CI/CD pipeline with some automation in the code.

Home

- [Gitlab Administration](#)
 - [Warnings](#)
 - [Groups and sub-groups](#)
 - [administration](#)
 - [course](#)
 - [student projects for a specific course](#)
 - [student projects setup](#)
 - [Automation](#)
 - [Creating subgroups](#)
 - [Steps](#)
 - [Forgetful students](#)
 - [Standard Gitlab groups](#)
 - [bachelor](#)
 - [msc](#)
 - [management](#)
 - [Naming conventions](#)
 - [General rules](#)
 - [Users](#)
 - [Course project naming](#)
 - [About student projects:](#)
 - [Scripts](#)

Gitlab Administration

This project is for Gitlab Administration and Management.

WORK in PROGRESS

Warnings

- gitlab server works fine for the workload that we have BUT, towards the end of each semester where most students actually use it at the same time, it works noticeably slower. I have identified the problem being in insufficient amount of RAM. Even though the VM only has 4 CPUs the CPU load seems not to top when the server is slow, it is rather related to the server running out of RAM and swapping heavily to disk. The server has only 8GB RAM.
- I've notified Lars Erik but he said that we cannot get more RAM :sadface:
- Perhaps **YOU** can notify your boss and help to lobby the level 4 management in providing some additional resources towards Lars Erik, such that the server gets more RAM.

Groups and sub-groups

Groups and subgroups allow hierarchical projects organisation such that logical groupings for maintenance and long-term archiving can be achieved. Below is a proposed scheme:

administration

Group for system administrators. Projects visible **ONLY** to system administrators.

course

- course - top-level group for all course-related projects
 - imt3673 - top-level group for all imt3673 related projects
 - 2018 - subgroup for archiving purposes for projects from that particular year. This forms a namespace for the concrete course projects
- course/imt3673/ The structure under the namespace, is up to a given course responsible. The structure might look as follows:
 - 2018/imt3673-lectures - Wiki, issue tracker for the lectures
 - 2018/imt3673-assignment1 - repo and sub-project for the course IMT3673 for assignment 1
 - 2018/imt3673-admin - for teachers only, administration tasks, etc.

student projects for a specific course

Note - generally, it is encouraged to follow that:

- student's private repo quota should not be used for course work
- course admins should setup internally space for student's submitted work, and grant access to that to groups/students

This way, it is much easier for course admins/lecturers to archive/cleanup the projects after the course, and, ensure limited visibility of student's work. Otherwise, student projects are shared publicly across years (e.g. Github) and there is no control over the projects after the course finishes.

student projects setup

Note - Gitlab's permissions flow from higher hierarchy levels downwards. It is important NOT TO GRANT permissions to students too high in the hierarchy. This has two important implications

- Student projects must sit outside of the general course materials to which all students in the course have access
- Student projects placeholder (top level) should NOT have "Request access" button enabled, and no student should be given access to that level. Student access should be granted to subgroups of the project placeholder.

Example

- course/imt3673/2018 - main top-level group to imt3673 in 2018. Students should not be given access at that level.
- course/imt3673/2018/imt3673-lectures - top level for lectures and lecture materials, students can request access and be granted access if you expect issue tracker/wiki/repo usage.
- course/imt3673/2018/projects - top-level student projects placeholder (ie. subgroup)
 - Students should not be given access at that level
 - That level is Internal
 - Deactivate "Request Access"
- course/imt3673/2018/projects/group_1 - a concrete group/or individual student project, with concrete assignment of roles to students.

Automation

Mariusz can run a script that generates the structure with the student usernames being used as the group names in this model automatically, and subsequently students can self-organise their projects in the hierarchy. The prerequisite is that all students are members of a well-defined Gitlab group or project.

Creating subgroups

For the automated setup to work, one needs to create two things – a subgroup for all the students subgroups, AND, a project, that ALL students become members of. This is necessary to generate the list of all Gitlab usernames that will source the automated process of generating the students subgroups.

Steps

- If you have something like `idatg2001-2022-ws` subgroup, then, make sure that this is made "Internal" and that NONE (but the teaching staff) is member to that subgroup, and, you disable "Request Access" to this subgroup (this subgroup must NOT have any members). The reason for this is that the Gitlab permissions flow "downwards" and it is impossible to prevent students to "see" each other work if they are given permission too high in the gitlab structure.
- The students should be granted **Developer** access to the course project.
- At the same level as your `idatg2001-2022-ws` group, you need to create a project, e.g. `idatg2001-2022`, and, you have to tell ALL students to REQUEST ACCESS to this project. We actually use that with Christopher for issue tracker about the course, and, for the Wiki, and for the repo to distribute example code to students.
- You can see how it looks in practice e.g.
 - <https://git.gvk.idi.ntnu.no/course/prog2006>
 - <https://git.gvk.idi.ntnu.no/course/prog2005>
- Once all your students are in your Course project (`idatg2001-2022`), then you can tell me "READY", and I will run a script, that will use the student usernames from your course project, to generate private subgroups to all your students in the designated group (`idatg2001-2022-ws`) automatically. The students will be made OWNERS of their respective subgroups, and they will be able to access that through the path to `idatg2001-2022-ws/<gitlabusername>`
- There are no restrictions on subgroups/projects, and students can manage their own space as they feel like. The subgroups are private, therefore, for peer-review or for collaborative group projects, the students (owners) must explicitly grant the permissions to their respective projects to open up their visibility.

Forgetful students

Students often "forget" to request access to your course project, and they are left out from the "workspace" – but, the script can be safely re-run multiple times, and, for already existing subgroups for existing students it skips doing anything, and it only adds the NEW students into the scheme.

Standard Gitlab groups

bachelor

For long-term storage of bachelor groups

- 2018
- 2019
 - Neodroid - for a particular bachelor group project team. The project's are all under the grouping, and can consist of multiple repositories (multiple projects in the Gitlab lingo).

msc

For long-term storage of master theses/projects. This is structurally equivalent to bachelor projects (see above). Note, this groups is called "MSc" in the UI, but the URL is with "msc". No CAPS in the URLs.

management

management - top level management repo for various management projects. We could have sub-groups for student-related management activities with student access (e.g. access to the game-lab issue tracker should probably not be under "teaching", but under management.

Naming conventions

General rules

- No CAPS in the URLs.
- Project name MATCHES exactly the GIT repo name (both, git repo and URL all lower case).
- Project name and Group names in the UI can use CamelCase or whatever seems appropriate.
- do not use underscore in names, use hyphen

Users

- usernames -
- name - RealName RealSurname
- bio - Student-Start-Year - ProgrammeName - StudentID

Course project naming

- *name* - use proper course code as the basis for the name. This way the projects are easily identifiable and have unique names. Always place projects in proper namespace, grouped accordingly as per "group/sub-groups" section above.

Conventions for specific iterations of a course as subgroups "CourseCode-YEAR"

About student projects:

Students (anyone) can create only 2 personal projects under their own namespace. This is to allow students to play with projects, have their own "fun" project, or to instantiate something that is subsequently moved under an appropriate namespace. It is a scrap-space for anyone.

All students should request access to a generic group: <https://git.gyk.idi.ntnu.no/students>

and then, subsequently, create and manage their own course-work related projects there, under that namespace. All temporary projects that relate to Host 2019 semester should go under student/2019. Anything that should live "longer" than a single semester, will need to be moved into one of the "permanent" project spaces

- research
- bachelor
- msc

They can request access to those groups, eg. "bachelor/2020" and "msc/2020" for next semester work. But, before that, we need to tidy up all the user bio's.

So, within the hierarchy, students can create "unlimited" projects. But, for "personal projects" they only have 2 slots. For any group work, or assignments, or student projects, they are, therefore, "forced" to use one of the predefined places for their projects in the "hierarchy".

Scripts

```
gitlab-rails runner "User.where(projects_limit: 10).each { |u| u.projects_limit = 0; u.save
```



C.2 Ivar Farup

Asked for privileged to quote in bachelor thesis, we will use to support our report - **ALLOWED**

Asked to use name - **ALLOWED**

Interview Ivar Farup

1. purpose of interview – high level about project

Group members presented thesis

2. ask opinion about existing LMS:

Do you have all files for a subject ready before start of subject?

Tasks are usually created and published whenever needed. Course compendia on the other hand is ready beforehand.

How do you inform students of new course material being published?

Usually I use the announcement feature of Blackboard. If something is vital I send mail as well which can be crossed off when creating announcements in Blackboard.

Does your LMS support for publishing course content at designated dates?

Not that I know of.

How often do you publish announcements?

Usually once a week

When sending an announcement, what do you want to accomplish?

For example:

- 1. Students should do something, read compendia...**
- 2. Just to be read by students, contains important information**
- 3. Inform of new/changed compendia**

In most cases my announcements are to inform students of a deliverable date or changes in task descriptions. Furthermore, when sending an announcement I want all students to deliver their tasks and accomplish as many students passing my subject.

What effect do your announcement have. Do you feel the announcements reach their target?

There are a multitude of students that does misses the announcements. For example, I can always expect I few follow up emails from individual students to explain the information further.

Are there any features you dislike in the LMS used by your organization?

There are not ways to separate or categorize announcements. For example, important announcements are often hard to find when there are a lot of announcements.

Not possible to draw out statistics or lists containing information about which students have delivered/passed and so forth.

What do you usually expect students to deliver in hand-ins?

Code on a repository

Writtenly

Multiple choice test integrated into Blackboard

How do you give feedback?

Usually we annotate on the original document and send back to students.

Furthermore, feedback on code can be given on git based systems.

Hand-ins in Blackboard Learn have a separate comment section that are useful for short comments on the delivered material.

Do you conduct peer-reviews?

Depends on the course. If possible yes, but usually peer-reviews are conducted with the student assistants.

3. present our solution

Website – git pages

Good impression of website design in developed system

Subject creation

Good impression of access control for courses in developed system

Content creation

Methods for teachers to publish contents in developed system was received in a good manner.

Announcement

Ivar Farup was positive to the release of announcements in developed system

Hand-ins

Not yet implemented, discussed possibilities

CI/CD - Automation

Good impression of possibilities with automations in course subjects

Overall Impression – Excellent tool for teachers, but needs to include hand-ins and other vital features to be used in courses.

C.3 Frank Alexander Kramer

Asked for privilege to quote in bachelor thesis, we will use to support our report -**ALLOWED, on basis Frank Alexander Kraemer can review before publishing**

Asked to use name – **ALLOWED, on basis Frank Alexander Kraemer can review before publishing**

Interview Frank Alexander Kraemer

Frank Alexander Kraemer is an Associate Professor and Deputy Head of Department, Education.

1. purpose of interview – high level about project

Group members presented thesis

2. ask opinion about existing LMS:

Frank Kramer elaborated on the discontent with Blackboard on a faculty level. The best case scenario for a LMS suited to IT knowledgeable teachers would be a system based on many technologies. A LMS that does some assumptions and enables teachers to create or remove packages as seen fit.

Why are people dissatisfied with modern LMS?

LMS are designed with features for all institutions. Instead, it should be based on open technology with a small layer on top to integrate them all. LMS should not make too many assumptions, but act as a framework to work from. Institutions could hire people to build around the service based on their needs instead of using an all-encompassing product which so far has not been a success.

How do you publish announcements?

One of the few functionalities Frank Alexander considers to be working well in Blackboard. Teachers have a rich text client where one can write and paste in anything to be announced. Can also choose to send announcement as mail.

As of now all announcements need to be published on Blackboard, a policy at NTNU. Furthermore, all course related information needs to be available at the same location of course subjects.

What are the effects of publishing announcements?

Unsure, never know if the announcements have been read or not. Should be able to have a commenting functionality for students as an arena to discuss the published information.

What do you expect students to deliver in a course?

Most commonly PDF documents, sometimes Word documents for peer review. Also create Panopto catalog for delivering videos.

Negative experiences with Blackboard

Frustrating to draw out who has delivered and not on any given task. For example, a text document could have been provided by the system as the information is often needed when documenting the progress and status of students in a course. Furthermore, Frank Alexander Kramer elaborated on the need for working offline with zipped files and metadata instead of needing to be online and using user interfaces that are limiting.

4. Discussing markdown as input

Frank and other mathematicians like markdown. Frequently used in “R” and used in python scripting and documentation.

5. Jekyll and transforming written documents into websites

Jekyll works well for creating websites at first. Despite this, Frank did not want to write in hascall but python instead. He therefore used his own version of “Pandoc” a commandline tool for document converting and saved a lot of time. The key idea of his own tool was to have different markup in the same file. Furthermore, there are not support for multicells in markdown and Frank found a markup language with the best support for cells and combined them into the document conversions tool.

C.4 Rune Hjelsvold

Asked for privileged to quote in bachelor thesis, we will use to support our report - **ALLOWED**

Asked to use name? If not only their quotes - **ALLOWED**

Interview Rune Hjelsvold 08.04.2022

1. purpose of interview – high level about project

Group members presented thesis.

2. ask opinion about existing LMS – user experience

Do you have all files for a subject ready before start of subject? Eller

Subject description and first few deliverables description are ready beforehand. The rest of course materials are published one week prior but for the most part created before the start of each semester. In addition the material is supplemented throughout the year

How do you inform students of new course material being published?

It depends. On a weekly basis, lectures and mentimeter happens regularly and is not given announcements when published. But if something out of the regular happens, for example an error was found in a task description or other critical information I send dedicated announcements in Blackboard and a mail.

Rune Hjelsvold often receives mail from student not knowing where the new material is published. It should be possible to link documents to announcements.

Rune Hjelsvold elaborated on the difficulties of informing students as there are a multitude of systems and nothing is centralized. It is important to involve and remind people throughout the first half of a semester how they work and get used to the workflow. As a consequence, the problems related to misinformed students falls of when they get used to the new methods of working.

Does your LMS support for publishing course content at designated dates?

Blackboard does allow for documents to be removed at a specified date, for example when a deadline has passed. But usually documents containing information should not be removed as they can be useful even afterwards of the deadline.

When sending an announcement, what do you want to accomplish?

For example:

- 1. Students should do something, read compendia...**
- 2. Just to be read by students, contains important information**
- 3. Inform of new/changed compendia**

Almost always announcements are meant to inform students of information. Assuring the message is received is therefore vital.

What effect do your announcement have. Do you feel the announcements reach their target?

Considering that a decent amount of time is often spent correcting students or elaborating on announcements the effects can be argued to have a lackluster effect. Despite this, my courses involved new methods for students to work than found in other subject. With this in mind, the students usually become used to the workflow after the initial semester and problems correcting or elaborating falls off.

Are there any features you dislike in the LMS used by your organization? Rune Hjelsvold described frustration towards LMS should help users archive, present and structure course materials. Despite this, there are no functionalities to directly link documents and elements within Blackboard. In practice, users end up reading one document and must navigate several layers to find corresponding documents referenced.

What do you usually expect students to deliver in hand-ins?

Code

Written documents

Multiple choice tests

3. present our solution

Website – git pages

Good impression of website design and layout

Subject creation

Rune Hjelsvold has prior experience with Mariusz Nowostawski user access control subject setup. The subject creation as described by students

Content creation

Rune Hjelsvold positively affirmed the workflow of teachers submitting their course materials through Git based commands.

Announcement

Positive to the announcements functionality in developed system

Hand-ins

Not yet implemented, discussed possibilities. Mentioned several courses that uses Gitlab or external platforms for delivering tasks by students.

CI/CD - Automations

Good impression of possibilities with automations in course subjects

Overall Impression

Excellent tool for teachers, but needs to include hand-ins and other vital features to be used in courses. In addition it is difficult utilizing a new system if not all of NTNU staff does aswell. There should be concurrency in the systems used throughout subjects.

C.5 Henrik Johnsen

Interview with Henrik Johnsen

What is your role, and how many on NTNU have the same role?

- Administrator of the IDI GitLab instance. “One and only”.
 - o Works as department Engineer/maintenance.
 - o The Institute are responsible for the instance; Henrik is informally the owner.

What can be told about the GitLab instance itself?

- “no strings attached”, everything is possible post-installment.
- No significant downtime, only vulnerable if the instance has not been updated in a long time.
 - o Updates are performed *manually*.
- Open for suggestions on improvements/implementations, as long as they are proven to be effective.

What runs the GitLab instance?

- GitLab is run on VMware, everything is virtualized. Simple as-you-go scaling possibilities. Approx. 10 minutes to scale up.
- wcpod.ntnu.no hosts about 150-200 production servers.
- As of 04.03.2022, 7767 users on the instance.
- 14TB of storage.
- Three instances: main, intern (used for testing) and dev (development)
- 6 Runners taking care of the CI/CD jobs
- Possible for the bachelor group to receive their own instance upon request.

What limitations are set for a GitLab user?

- The limit is at 10 repos per user as standard, however special users such as lecturers have another quota. No limitation on *size* per repo, though Henrik monitors the repo sizes. Maximum import size at 50MB.
 - o Three kinds of users: Student, Teacher and Bots. Bots can be used to automate processes or solve some issues.

- No policies on what pictures to publish, though a manual check is done should it exceed 1GB. Henrik has an Admin Dashboard to monitor the GitLab activity.
- Git Integrations also open for implementation on the instance.
 - o Mattermost, identical to Slack, has been implemented on the instance.

What security measures have been implemented?

- Cisco AMP is installed on the instance, additional tool to secure the instance against malicious files.
- 2FA has been activated as per NTNU SOC's request.
 - o The SOC also asks for additional support in form of logs, should an investigation take place.
- All accounts are valuable: Anyone can be used as an attack surface/proxy for further infiltration of the system.
- No SLAs to be concerned about.

D Project Management

D.1 Time log

D.1.1 Nicholas Bodvin Sellevåg

Employee Timecard

Employee Name: Nicholas B.S E-mail: _____ Year to date totals:
 Manager: _____ Phone: _____ Regular hrs: **329.5** Overtime hrs: **0** Total: **329.5**

January, February, March Employee Timecard: Daily, Weekly, Monthly, Yearly

January	Week 1	Overtime	Week 2	Overtime	Week 3	Overtime	Week 4	Overtime	Week 5	Overtime
Monday										
Tuesday									1	
Wednesday									2.5	
Thursday							4		1	
Friday							3		3	
Saturday										
Sunday										
Total weekly hours	0	0	0	0	0	0	7	0	7.5	0
Jan. total: Regular hours	14.5		Jan. total: Overtime		0					

February	Week 1	Overtime	Week 2	Overtime	Week 3	Overtime	Week 4	Overtime	Week 5	Overtime
Monday	4		4.5		3		4			
Tuesday	3		3		2		2			
Wednesday	5		4		1					
Thursday	4		10.5		3		5			
Friday			5				4			
Saturday										
Sunday										
Total weekly hours	16	0	27	0	9	0	15	0	0	0
Feb. total: Regular hours	67		Feb. total: Overtime		0					

March	Week 1	Overtime	Week 2	Overtime	Week 3	Overtime	Week 4	Overtime	Week 5	Overtime
Monday	1		5		6		2			
Tuesday	5				12		5			
Wednesday			4		4		6			
Thursday					3		4			
Friday	6		3		2		4			
Saturday										
Sunday					6					
Total weekly hours	12	0	12	0	33	0	21	0	0	0
Mar. total: Regular hours	78		Mar. total: Overtime		0					

Employee Timecard

Employee Name: Nicholas B.S E-mail: _____
 Manager: _____ Phone: _____

Year to date totals:
 Regular hrs: **329.5** Overtime hrs: **0** Total: **329.5**

April, May, June Employee Timecard: Daily, Weekly, Monthly, Yearly

April	Week 1	Overtime	Week 2	Overtime	Week 3	Overtime	Week 4	Overtime	Week 5	Overtime
Monday	5		5		5		5			
Tuesday	5		6		4		4			
Wednesday	3		4		4		4			
Thursday	4		3		5		5			
Friday	5		7		6		6			
Saturday										
Sunday										
Total weekly hours	22	0	25	0	24	0	24	0	0	0
Apr. total: Regular hours	95		Apr. total: Overtime		0					

May	Week 1	Overtime	Week 2	Overtime	Week 3	Overtime	Week 4	Overtime	Week 5	Overtime
Monday	5		5		4					
Tuesday	6		5							
Wednesday	5		5		9					
Thursday	5		5		9					
Friday	7		5							
Saturday										
Sunday										
Total weekly hours	28	0	25	0	22	0	0	0	0	0
May total: Regular hours	75		May total: Overtime		0					

D.1.2 Yan Senko

Employee Timecard

Employee Name: Yan Senko E-mail: _____ Year to date totals:
 Manager: Nicholas Phone: _____ Regular hrs: **248.5** Overtime hrs: **0** Total: **248.5**

January, February, March Employee Timecard: Daily, Weekly, Monthly, Yearly

January	Week 1	Overtime	Week 2	Overtime	Week 3	Overtime	Week 4	Overtime	Week 5	Overtime
Monday										
Tuesday									1	
Wednesday									2.5	
Thursday									1	
Friday							1		3	
Saturday										
Sunday										
Total weekly hours	0	0	0	0	0	0	1	0	7.5	0
Jan. total: Regular hours	8.5		Jan. total: Overtime		0					

February	Week 1	Overtime	Week 2	Overtime	Week 3	Overtime	Week 4	Overtime	Week 5	Overtime
Monday			3		4.5		7			
Tuesday	2				1		2			
Wednesday	7		1		2					
Thursday					3					
Friday					2.5					
Saturday										
Sunday										
Total weekly hours	9	0	4	0	13	0	9	0	0	0
Feb. total: Regular hours	35		Feb. total: Overtime		0					

March	Week 1	Overtime	Week 2	Overtime	Week 3	Overtime	Week 4	Overtime	Week 5	Overtime
Monday	1		1		6		4		5	
Tuesday	5				5		5		1	
Wednesday	4				6		6			
Thursday							2			
Friday	6		3				2			
Saturday							2			
Sunday										
Total weekly hours	16	0	4	0	17	0	21	0	6	0
Mar. total: Regular hours	64		Mar. total: Overtime		0					

Employee Timecard

Employee Name: Yan Senko E-mail: _____
 Manager: Nicholas Phone: _____

Year to date totals:
 Regular hrs: **248.5** Overtime hrs: **0** Total: **248.5**

April, May, June Employee Timecard: Daily, Weekly, Monthly, Yearly

April	Week 1	Overtime	Week 2	Overtime	Week 3	Overtime	Week 4	Overtime	Week 5	Overtime
Monday	2		5		1.5		1			
Tuesday	3		2		1.5		5			
Wednesday	3		3		1		4			
Thursday	4		2		3		7			
Friday	5		7		3		4			
Saturday										
Sunday					2					
Total weekly hours	17	0	19	0	9	0	21	0	0	0
Apr. total: Regular hours	66		Apr. total: Overtime		0					

May	Week 1	Overtime	Week 2	Overtime	Week 3	Overtime	Week 4	Overtime	Week 5	Overtime
Monday	5		5		4					
Tuesday	6		5							
Wednesday	5		5		9					
Thursday	5		5		9					
Friday	7		5							
Saturday										
Sunday										
Total weekly hours	28	0	25	0	22	0	0	0	0	0
May total: Regular hours	75		May total: Overtime		0					

D.1.3 Fabian Kongelf

Employee Timecard

Employee Name: Fabian Kongel E-mail: _____ Year to date totals:
 Manager: Nicholas Phone: _____ Regular hrs: 344 Overtime hrs: 0 Total: 344

January, February, March Employee Timecard: Daily, Weekly, Monthly, Yearly

January	Week 1	Overtime	Week 2	Overtime	Week 3	Overtime	Week 4	Overtime	Week 5	Overtime
Monday										
Tuesday										1
Wednesday										3
Thursday							4			1
Friday							3			3
Saturday										
Sunday										
Total weekly hours	0	0	0	0	0	0	7	0	8	0
Jan. total: Regular hours	15		Jan. total: Overtime		0					

February	Week 1	Overtime	Week 2	Overtime	Week 3	Overtime	Week 4	Overtime	Week 5	Overtime
Monday	4		4		3		1		3	
Tuesday	4		3		2		2			
Wednesday	7		4		1		2			
Thursday	4		3		3		3			
Friday			5		3		5			
Saturday										
Sunday										
Total weekly hours	19	0	19	0	12	0	13	0	3	0
Feb. total: Regular hours	66		Feb. total: Overtime		0					

March	Week 1	Overtime	Week 2	Overtime	Week 3	Overtime	Week 4	Overtime	Week 5	Overtime
Monday			5		6		2		5	
Tuesday	5				5		5		7	
Wednesday	4		3		6		6		1	
Thursday	5				3		1		4	
Friday	6		3		4		1			
Saturday							2		2	
Sunday										
Total weekly hours	20	0	11	0	24	0	17	0	19	0
Mar. total: Regular hours	91		Mar. total: Overtime		0					

Employee Timecard

Employee Name: Fabian Kongel E-mail: _____
 Manager: Nicholas Phone: _____

Year to date totals:
 Regular hrs: 344 Overtime hrs: 0 Total: 344

April, May, June Employee Timecard: Daily, Weekly, Monthly, Yearly

April	Week 1	Overtime	Week 2	Overtime	Week 3	Overtime	Week 4	Overtime	Week 5	Overtime
Monday			5				5		4	
Tuesday	2		6		1		7		4	
Wednesday	3		7		5		6		4	
Thursday	4				2		5		4	
Friday	5		5		1		4		4	
Saturday			4							
Sunday										
Total weekly hours	14	0	27	0	9	0	27	0	20	0
Apr. total: Regular hours	97		Apr. total: Overtime		0					

May	Week 1	Overtime	Week 2	Overtime	Week 3	Overtime	Week 4	Overtime	Week 5	Overtime
Monday	5		5		4					
Tuesday	6		5							
Wednesday	5		5		9					
Thursday	5		5		9					
Friday	7		5							
Saturday										
Sunday										
Total weekly hours	28	0	25	0	22	0	0	0	0	0
May total: Regular hours	75		May total: Overtime		0					

D.1.4 Oddbjørn S. Borge-Jensen

Employee Timecard

Employee Name: Oddbjørn S.BJ E-mail: _____ Year to date totals:
 Manager: Nicholas Phone: _____ Regular hrs: 309 Overtime hrs: 0 Total: 309

January, February, March Employee Timecard: Daily, Weekly, Monthly, Yearly

January	Week 1	Overtime	Week 2	Overtime	Week 3	Overtime	Week 4	Overtime	Week 5	Overtime
Monday										
Tuesday										1
Wednesday										2.5
Thursday							4			1
Friday							3			3
Saturday										
Sunday										
Total weekly hours	0	0	0	0	0	0	7	0	7.5	0
Jan. total: Regular hours	14.5		Jan. total: Overtime		0					

February	Week 1	Overtime	Week 2	Overtime	Week 3	Overtime	Week 4	Overtime	Week 5	Overtime
Monday	4		3				4			
Tuesday	3		3		2		2			
Wednesday	5		4		1		5			
Thursday	4		6		3		4			
Friday			5		2.5		2			
Saturday					3					
Sunday					1					
Total weekly hours	16	0	21	0	12.5	0	17	0	0	0
Feb. total: Regular hours	66.5		Feb. total: Overtime		0					

March	Week 1	Overtime	Week 2	Overtime	Week 3	Overtime	Week 4	Overtime	Week 5	Overtime
Monday	1				6					
Tuesday	5		2		5		5			
Wednesday	3		4		3		4			
Thursday	3		1		2		5			
Friday	6		3		5		6			
Saturday										
Sunday										
Total weekly hours	18	0	10	0	21	0	20	0	0	0
Mar. total: Regular hours	69		Mar. total: Overtime		0					

Employee Timecard

Employee Name: Oddbjørn S.BJ E-mail: _____
 Manager: Nicholas Phone: _____

Year to date totals:
 Regular hrs: **309** Overtime hrs: **0** Total: **309**

April, May, June Employee Timecard: Daily, Weekly, Monthly, Yearly

April	Week 1	Overtime	Week 2	Overtime	Week 3	Overtime	Week 4	Overtime	Week 5	Overtime
Monday	5		5		3		2			
Tuesday	2		5		5		4			
Wednesday	3		7		4		4			
Thursday	4		2		6		6			
Friday	5		5		1		6			
Saturday										
Sunday										
Total weekly hours	19	0	24	0	19	0	22	0	0	0
Apr. total: Regular hours	84		Apr. total: Overtime		0					

May	Week 1	Overtime	Week 2	Overtime	Week 3	Overtime	Week 4	Overtime	Week 5	Overtime
Monday	5		5		4					
Tuesday	6		5							
Wednesday	5		5		9					
Thursday	5		5		9					
Friday	7		5							
Saturday										
Sunday										
Total weekly hours	28	0	25	0	22	0	0	0	0	0
May total: Regular hours	75		May total: Overtime		0					

D.2 Project Planning

Project plan – Git da gitt!

Nicholas Bodvin Sellevåg

Nicholbs@stud.ntnu.no

Oddbjørn Siver Borge-Jensen

oddbjosb@stud.ntnu.no

Fabian Kongelf

Fabiako@stud.ntnu.no

Yan Senko

Yans@stud.ntnu.no

Goals	3
Assignment	3
Background	3
Project goals.....	3
Scope.....	4
Background information.....	4
Limitations	4
Project organization.....	5
Roles and responsibilities	5
Routines and rules	5
Planning.....	5
Group working method and process	5
Scientific methodology.....	6
Quality Assurance.....	6
Resources.....	6
Produced documents	7
Tools.....	7
Progress plan.....	7
Sources.....	7

Goals

Assignment

Neither students nor teachers are satisfied with existing LMS (Learning Management Systems¹). LMS's should solve the main challenges of information sharing, information classification (public/private), process submissions, feedback deliverance, conduct digital tests, and provide a discussion forum.

The client wishes to determine whether it is possible to fulfill the needs traditionally met by LMS software through other systems already in use at NTNU such as Git and Inspira. For Git, the goal will be to fulfill the requirements using GitLab Pages specifically, as earlier attempts at utilizing pure Git has caused confusion among students due to the overwhelming UI.

Our assignment is to look at the parts of the system that could be replaced by Git. As provided by the client, the key functionalities to be explored are:

1. Easy access to public information
2. Easy access to private information
3. Make announcements
4. Receive submissions and provide feedback

Background

NTNU's contract with their current LMS, Blackboard, will soon reach its end. Given dissatisfaction with this platform from both students and teachers, NTNU wants to investigate alternative solutions for the services usually provided through an LMS. Git has already been used as a limited replacement, but the complex nature of the site resulted in some students struggling with how to use it and access all the information they needed.

To amend this while still preserving the strengths of having the subject hosted on a git repository, our client Erik Hjelmås wants to explore the possibility of setting up a website presenting the information in an orderly manner that's easy to navigate. The suggested solution is to achieve this through GitLab Pages.

We are a group of four students from Digital Infrastructure and Cybersecurity at NTNU. Through our time studying here we have become familiar with the use and purposes of Blackboard, as well as attained a considerable amount of experience with designing user friendly websites. While this project will not entail creating and hosting a site entirely from scratch, we believe that much of the knowledge we have acquired will be beneficial for the completion of the assignment.

Project goals

What we want to achieve with this bachelor project.

Result goals

- Produce a git repository that can easily deploy to a website using GitLab Pages
- Determine an efficient method of displaying public information on a git Page
- Determine an efficient method of displaying private information on a git Page
- Explore methods of broadcasting announcements through git
- Explore methods of receiving submission and providing feedback through git

¹ Example of this are Blackboard and Itslearning.

Learning goals

- Strengthen our ability to plan and execute a project using agile methodologies
- Utilize and build upon the knowledge and experience acquired through our years of studying at NTNU
- Attain in-depth knowledge of git and GitLab Pages

Scope

Background information

A learning management system (LMS) is a software application for the administration, documentation, tracking, reporting, automation, and delivery of educational courses, training programs, or learning and development programs.

An LMS is a platform for online content, primarily learning materials and a support tool for teachers and students to best teach and follow their courses. Teachers can publish a subject's curriculum, syllabus, answer sheets, results, post feedback and create online tests. However, a lot of these systems are similar and do not allow for customization and personalization. If a school or a subject requires a specific way of teaching, they would have to compromise on their vision to fit their ideal structure within a system that is not made to support it.

Git is a cloud-based service offering cloud storage of repositories. A lot of teachers, especially within the IT sector, use git repositories to store and share their learning materials. One Git implementation offers a service called GitLab Pages which allows a Git user to create a webpage of their git-repo's content. What if a teacher can simply convert their Git-repo content into a webpage structure instead of a file structure? A Git web structure might be able to offer the same features as a conventional LMS but with the customization Git-repos possess.

Limitations

The assignment provided by the client defines specific functionalities that should be explored, and as such any other functionalities provided by Blackboard that could theoretically necessitate reproducing will not be explored in our thesis.

In order not to make the thesis about how to create the perfect website we are limiting the design interface to a standard computer screen with the dimension of 16:9 and an average smartphone screen dimension.

A request of the client was to use as simple and self-produced code as possible to avoid security issues with imported code and use of well-known frameworks, as well as updates to imported code may cause issues for the rest of the application.

The client explicitly does not wish for a technically advanced solution and prefers a simple and efficient site that does not introduce unnecessary security issues or other potential symptoms of over-engineering.

As the main motivation behind moving to GitLab Pages is its ease of use compared to the Git repository itself, solutions should not require more than a base level of git knowledge to use.

Project organization

Roles and responsibilities

Nicholas has work experience with SCRUM, therefore he is the Scrum master and will be responsible for managing the Scrum board, setup and arranging stand-up meetings and sprint-reviews, and lastly make sure the group members assigned tasks are completed.

All group members are regarded as developers. As a baseline, each member works on delegated tasks and is responsible for updating their status in the task management tools decided upon. Moreover, developers are all encompassing in contrast to delegating specific tasks to specific members - e.g. two people are responsible for research and two are developers - members will move between tasks as deemed necessary. This both prevents a member from being stuck with a task they do not enjoy and losing motivation and allows us to prioritize research over development or vice versa at any given time.

Should the need for other responsibilities arise will they be delegated to a group member based on the members' individual capabilities and their current workload.

Routines and rules

All group members are expected to work at least 16 hours a week. The group can agree to amend this requirement for a specific group member should there be extraneous circumstances like work or sickness that prevent them from dedicating the amount of work expected.

If it becomes clear that the group needs to increase the mandatory workload to finish the project in time with a satisfactory level of quality, the minimum requirement can be increased to a maximum 30 hours a week.

These are just minimum limits to prevent the project from stagnating. Members are expected to put in an average amount of work per week considerably higher than the minimum requirement.

Meetings outside of the usual stand-ups and sprint reviews should be planned at least 24 hours in advance and have a clear agenda. Repeated failure to show up to these and any other kind of meeting and a refusal to communicate with the rest of the group will be communicated to NTNU.

Meetings will be held with the client every other week to give their opinion on the current state of the product and ask clarifying questions that we might have about the upcoming functionalities. In addition, Guoqiang Li will be helping us ensure the quality of the bachelor thesis through weekly meetings. Relevant subjects include working methodology, current progress, and ensuring a scientific basis for decision and conclusion made.

Planning

Group working method and process

SCRUM is used as the framework for our working methodology. The choice of SCRUM was based on its focus on iterative development and continual improvement.

Firstly, group members work iteratively throughout Sprints. Bachelor project lasts for five months and have been divided into eight two-week periods named as Sprints. The Scum-team work together to finish designated tasks within each Sprint. Moreover, all processes within the bachelor project takes place within the Sprint. To illustrate, Sprint Planning, Sprint Review, Sprint Retrospective and Daily Scrum are all part of each Sprint. A new Sprint is started immediately following the termination

of another. Each Sprint starts with determining the goals and tasks to be completed during the Sprint Planning phase.

Secondly, Daily Scrum enhances communication and collaboration within the team. Group members attend two days a week for 15 minutes to review the progress of their delegated tasks. The Daily Scrum is also an arena for members to obtain feedback and help from others.

Thirdly, results from Sprints are presented during Sprint Reviews alongside future adaptations. The process includes all actors within the project and participants present their accomplishments. Future Sprints are based on the results of tasks and remaining items in "Product Backlog", a prioritized catalog of work not yet started.

Lastly, Sprint Retrospectives ensures continual improvement of teamwork. The group reflects on the terminated Sprint's positive and negative aspects. In practice, topics include from interactions between team members, tools used during development, implementation of Scrum processes and definition of done for elements within the "Product Backlog" that took part during the Sprint.

Scientific methodology

The client wants to explore possibilities of using other existing systems already in use to solve the core challenges of LMS. Initially, Gitlab is the primary target to survey. Gitlab is not as a standard configured to address the needs of an LMS.

For this project we plan to consciously adhere to the working methods of the scientific method. Throughout our time at NTNU we have worked on several projects using agile methodologies, which are similar in nature to the ideas behind the scientific method and will be used for this project as well. For each functionality proposed by the client we will have a research phase to determine what criteria must be met to be able to consider the functionality complete, and what tools we have at our disposal to fulfill these criteria using GitLab Pages. These criteria will be gathered to make up a working Definition of Done for the functionality, which is prone to change should further observations reveal missing or flawed criteria.

Each potential solution will serve as a hypothesis, and we will conduct an experiment in the form of user acceptance testing and if necessary, a comparative analysis based on the criteria we have defined and the functionality of the existing solutions in Blackboard. Should the solution not meet our criteria, we will use the observations made to produce a new solution. If we are unable to find any satisfactory solutions to one of the assignment functionalities, we will perform an analysis of the criteria of this functionality and the git tools we have attempted to use. The purpose of this analysis would be to detail why we found that git might not be a suitable tool for that specific job.

When the alternative system has been implemented, the thesis will include a comparative analysis between our solution and its existing Blackboard counterpart. The analysis is based on the functionalities contrived to address the core challenges of LMS in existing and alternative systems.

Quality Assurance

Resources

All documents – including but not limited to time logs, the project plan, and the report – will be hosted on Microsoft Teams.

Resources used within the group that are not part of the finished project (e.g. links and references) will be stored on the group's Discord channel.

Produced documents

Bachelor project includes software development and configuration of Gitlab. To ensure we fulfill the requirements of the task in the way the client expects, we will work together with the client to create a definition of done for each key functionality. Following an iteration of a task, the result will be tested and its performance in accord with the definition of done will be reported in respective “functionality test” documents. If the results are satisfactory that task will be deemed complete, and we will move on to the next.

It is noteworthy that all documents will adhere to Harvard’s rule of citation.

Tools

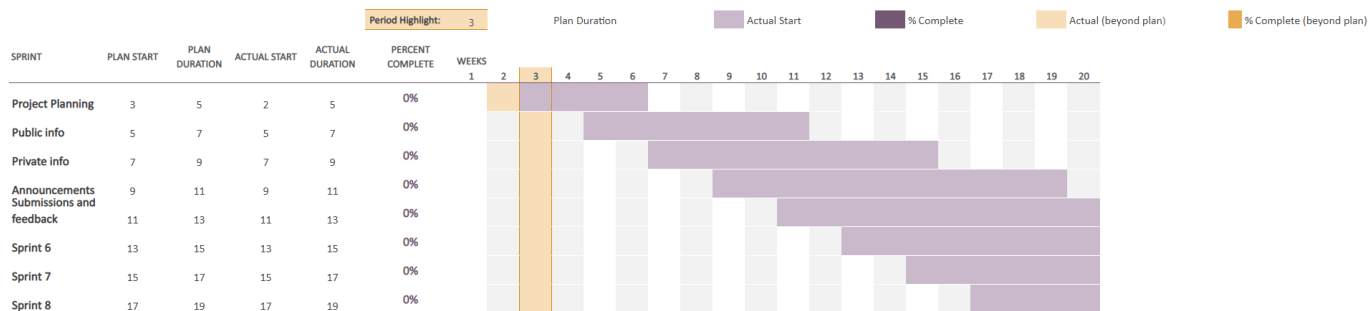
- Microsoft Teams for communication with client and guidance counselor as well as writing shared documents
- Git
- GitLab Pages
- Visual Studio Code for any programming – mainly HTML and CSS
- Discord for internal meetings
- Jira - team management and issue tracking

Progress plan

Milestones for the project are listed in Gantt Chart below. Some Sprints are named in accordance with a milestone to reach for Gitlab development. For example, “Public info” entails the duration to develop public accessible information for members of subjects through Gitlab. If the functionality requires additional development time, the Sprint can be prolonged or have its tasks relocate to another sprint.

The contents of the last few sprints have not yet been determined but will be allocated in the future as we find out what needs to be done.

Git da Git



To summarize, the team's progress is tracked through the individual functionalities implemented through Gitlab.

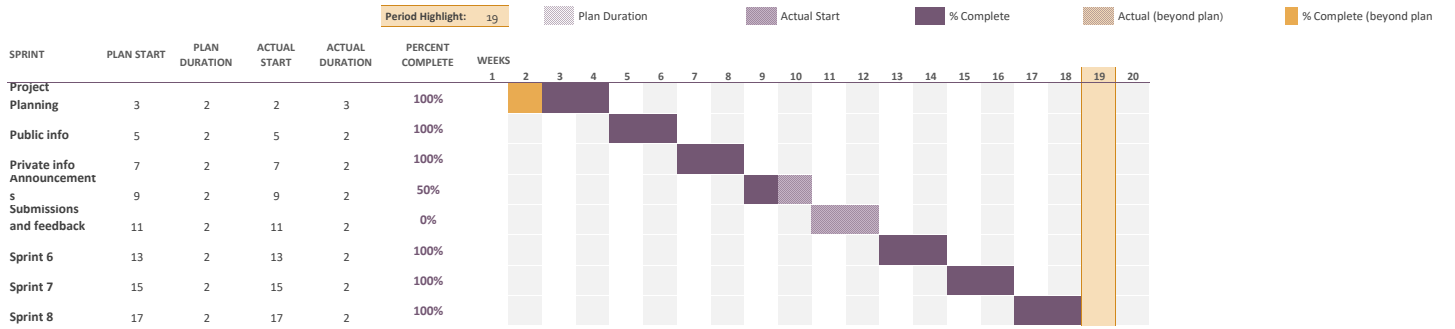
Sources

Wikipedia (2022) *Git*, available from URL: <https://en.wikipedia.org/wiki/Git> (read: 20th of January 2022)

Wikipedia (2022) *Learning management system*, available from URL:
https://en.wikipedia.org/wiki/Learning_management_system (read: 20th of January 2022)

D.3 Gantt Chart

Git da Git



D.4 Project contract

Fastsatt av prorektor for utdanning 10.12.2020

STANDARDAVTALE

om utføring av studentoppgave i samarbeid med ekstern virksomhet

Avtalen er ufravikelig for studentoppgaver (heretter oppgave) ved NTNU som utføres i samarbeid med ekstern virksomhet.

Forklaring av begrep

Opphavsrett

Er den rett som den som skaper et åndsverk har til å fremstille eksemplarer av åndsverket og gjøre det tilgjengelig for allmennheten. Et åndsverk kan være et litterært, vitenskapelig eller kunstnerisk verk. En studentoppgave vil være et åndsverk.

Eiendomsrett til resultater

Betyr at den som eier resultatene bestemmer over disse. Utgangspunktet er at studenten eier resultatene fra sitt studentarbeid. Studenten kan også overføre eiendomsretten til den eksterne virksomheten.

Bruksrett til resultater

Den som eier resultatene kan gi andre en rett til å bruke resultatene, f.eks. at studenten gir NTNU og den eksterne virksomheten rett til å bruke resultatene fra studentoppgaven i deres virksomhet.

Prosjektbakgrunn

Det partene i avtalen har med seg inn i prosjektet, dvs. som vedkommende eier eller har rettigheter til fra før og som brukes i det videre arbeidet med studentoppgaven. Dette kan også være materiale som tredjepersoner (som ikke er part i avtalen) har rettigheter til.

Utsatt offentliggjøring

Betyr at oppgaven ikke blir tilgjengelig for allmennheten før etter en viss tid, f.eks. før etter tre år. Da vil det kun være veileder ved NTNU, sensorene og den eksterne virksomheten som har tilgang til studentarbeidet de tre første årene etter at studentarbeidet er innlevert.

1. Avtaleparter

Norges teknisk-naturvitenskapelige universitet (NTNU) Institutt: Institutt for informasjonssikkerhet og kommunikasjonsteknologi
Veileder ved NTNU: Guoqiang Li
Virksomhet: NTNU Virksomhet sin kontaktperson, e-post og tlf.: Erik Hjelmås erik.hjelmas@ntnu.no 93034446
Student: Nicholas Bodvin Sellevåg Fødselsdato: 150798
Ev. flere studenter ¹ Yan Senko 121001 Fabian Kongelf 190996 Oddbjørn S. Borge-Jensen 110400

Partene har ansvar for å klarere eventuelle immaterielle rettigheter som studenten, NTNU, den eksterne eller tredjeperson (som ikke er part i avtalen) har til prosjektbakgrunn før bruk i forbindelse med utførelse av oppgaven. Eierskap til prosjektbakgrunn skal fremgå av eget vedlegg til avtalen der dette kan ha betydning for utførelse av oppgaven.

2. Utførelse av oppgave

Studenten skal utføre: (sett kryss)

Masteroppgave	<input type="checkbox"/>
---------------	--------------------------

¹ Dersom flere studenter skriver oppgave i fellesskap, kan alle føres opp her. Rettigheter ligger da i fellesskap mellom studentene. Dersom ekstern virksomhet i stedet ønsker at det skal inngås egen avtale med hver enkelt student, gjøres dette.

Bacheloroppgave	X
Prosjektoppgave	
Annen oppgave	

Startdato:
Sluttdato:

Oppgavens arbeidstittel er:
Git da gitt!

Ansvarlig veileder ved NTNU har det overordnede faglige ansvaret for utforming og godkjenning av prosjektbeskrivelse og studentens læring.

3. Ekstern virksomhet sine plikter

Ekstern virksomhet skal stille med en kontaktperson som har nødvendig faglig kompetanse til å gi studenten tilstrekkelig veiledning i samarbeid med veileder ved NTNU. Ekstern kontaktperson fremgår i punkt 1.

Formålet med oppgaven er studentarbeid. Oppgaven utføres som ledd i studiet. Studenten skal ikke motta lønn eller lignende godtgjørelse fra den eksterne for studentarbeidet. Utgifter knyttet til gjennomføring av oppgaven skal dekkes av den eksterne. Aktuelle utgifter kan for eksempel være reiser, materialer for bygging av prototyp, innkjøp av prøver, tester på lab, kjemikalier. Studenten skal klarere dekning av utgifter med ekstern virksomhet på forhånd.

Ekstern virksomhet skal dekke følgende utgifter til utførelse av oppgaven:

Dekning av utgifter til annet enn det som er oppført her avgjøres av den eksterne underveis i arbeidet.

4. Studentens rettigheter

Studenten har opphavsrett til oppgaven². Alle resultater av oppgaven, skapt av studenten alene gjennom arbeidet med oppgaven, eies av studenten med de begrensninger som følger av punkt 5, 6 og 7 nedenfor. Eiendomsretten til resultatene overføres til ekstern virksomhet hvis punkt 5 b er avkrysset eller for tilfelle som i punkt 6 (overføring ved patenterbare oppfinnelser).

² Jf. Lov om opphavsrett til åndsverk mv. av 15.06.2018 § 1

I henhold til lov om opphavsrett til åndsverk beholder alltid studenten de ideelle rettigheter til eget åndsverk, dvs. retten til navngivelse og vern mot krenkende bruk.

Studenten har rett til å inngå egen avtale med NTNU om publisering av sin oppgave i NTNUs institusjonelle arkiv på Internett (NTNU Open). Studenten har også rett til å publisere oppgaven eller deler av den i andre sammenhenger dersom det ikke i denne avtalen er avtalt begrensninger i adgangen til å publisere, jf. punkt 8.

5. Den eksterne virksomheten sine rettigheter

Der oppgaven bygger på, eller videreutvikler materiale og/eller metoder (prosjektbakgrunn) som eies av den eksterne, eies prosjektbakgrunnen fortsatt av den eksterne. Hvis studenten skal utnytte resultater som inkluderer den eksterne sin prosjektbakgrunn, forutsetter dette at det er inngått egen avtale om dette mellom studenten og den eksterne virksomheten.

Alternativ a) (sett kryss) Hovedregel

<input type="checkbox"/>	Ekstern virksomhet skal ha bruksrett til resultatene av oppgaven
--------------------------	--

Dette innebærer at ekstern virksomhet skal ha rett til å benytte resultatene av oppgaven i egen virksomhet. Retten er ikke-eksklusiv.

Alternativ b) (sett kryss) Unntak

<input type="checkbox"/>	Ekstern virksomhet skal ha eiendomsretten til resultatene av oppgaven og studentens bidrag i ekstern virksomhet sitt prosjekt
--------------------------	---

Begrunnelse for at ekstern virksomhet har behov for å få overført eiendomsrett til resultatene:

6. Godtgjøring ved patenterbare oppfinnelser

Dersom studenten i forbindelse med utførelsen av oppgaven har nådd frem til en patenterbar oppfinnelse, enten alene eller sammen med andre, kan den eksterne kreve retten til oppfinnelsen overført til seg. Dette forutsetter at utnyttelsen av oppfinnelsen faller inn under den eksterne sitt virksomhetsområde. I så fall har studenten krav på rimelig godtgjøring. Godtgjøringen skal fastsettes i samsvar med arbeidstakeroppfinnelsesloven § 7. Fristbestemmelsene i § 7 gis tilsvarende anvendelse.

7. NTNU sine rettigheter

De innleverte filer av oppgaven med vedlegg, som er nødvendig for sensur og arkivering ved NTNU, tilhører NTNU. NTNU får en vederlagsfri bruksrett til resultatene av oppgaven, inkludert vedlegg til denne, og kan benytte dette til undervisnings- og forskningsformål med de eventuelle begrensninger som fremgår i punkt 8.

8. Utsatt offentliggjøring

Hovedregelen er at studentoppgaver skal være offentlige.

Sett kryss

<input checked="" type="checkbox"/>	Oppgaven skal være offentlig
-------------------------------------	------------------------------

I særlige tilfeller kan partene bli enige om at hele eller deler av oppgaven skal være undergitt utsatt offentliggjøring i maksimalt tre år. Hvis oppgaven unntas fra offentliggjøring, vil den kun være tilgjengelig for student, ekstern virksomhet og veileder i denne perioden. Sensurkomiteen vil ha tilgang til oppgaven i forbindelse med sensur. Student, veileder og sensorer har taushetsplikt om innhold som er unntatt offentliggjøring.

Oppgaven skal være underlagt utsatt offentliggjøring i (sett kryss hvis dette er aktuelt):

Sett kryss

Sett dato

Sett kryss	Sett dato
<input type="checkbox"/>	ett år
<input type="checkbox"/>	to år
<input type="checkbox"/>	tre år

Behovet for utsatt offentliggjøring er begrunnet ut fra følgende:

Dersom partene, etter at oppgaven er ferdig, blir enig om at det ikke er behov for utsatt offentliggjøring, kan dette endres. I så fall skal dette avtales skriftlig.

Vedlegg til oppgaven kan unntas ut over tre år etter forespørsel fra ekstern virksomhet. NTNU (ved instituttet) og student skal godta dette hvis den eksterne har saklig grunn for å be om at et eller flere vedlegg unntas. Ekstern virksomhet må sende forespørsel før oppgaven leveres.

De delene av oppgaven som ikke er undergitt utsatt offentliggjøring, kan publiseres i NTNUs institusjonelle arkiv, jf. punkt 4, siste avsnitt. Selv om oppgaven er undergitt utsatt offentliggjøring, skal ekstern virksomhet legge til rette for at studenten kan benytte hele eller deler av oppgaven i forbindelse med jobbsøknader samt videreføring i et master- eller doktorgradsarbeid.

9. Generelt

Denne avtalen skal ha gyldighet foran andre avtaler som er eller blir opprettet mellom to av partene som er nevnt ovenfor. Dersom student og ekstern virksomhet skal inngå avtale om konfidensialitet om det som studenten får kjennskap til i eller gjennom den eksterne virksomheten, kan NTNUs standardmal for konfidensialitetsavtale benyttes.

Den eksterne sin egen konfidensialitetsavtale, eventuell konfidensialitetsavtale den eksterne har inngått i samarbeidprosjekter, kan også brukes forutsatt at den ikke inneholder punkter i motstrid med denne avtalen (om rettigheter, offentliggjøring mm). Dersom det likevel viser seg at det er motstrid, skal NTNUs standardavtale om utføring av studentoppgave gå foran. Eventuell avtale om konfidensialitet skal vedlegges denne avtalen.

Eventuell uenighet som følge av denne avtalen skal søkes løst ved forhandlinger. Hvis dette ikke fører frem, er partene enige om at tvisten avgjøres ved voldgift i henhold til norsk lov. Tvisten avgjøres av sorenskriveren ved Sør-Trøndelag tingrett eller den han/hun oppnevner.

Denne avtale er signert i fire eksemplarer hvor partene skal ha hvert sitt eksemplar. Avtalen er gyldig når den er underskrevet av NTNU v/instituttleder.

Signaturer:

Instituttleder:	
Dato:	
Veileder ved NTNU:	
Dato:	
Ekstern virksomhet:	
Dato:	
Student:	01/02/2022 <i>Nicholas Sellevåg</i> Nicholas Sellevåg
Dato:	
Ev. flere studenter	Yan Senko, 01/02/2022 <i>[Signature]</i>
	Fabian Kongelf, 01/02/2022 <i>Fabian Kongelf</i>
	Oddbjørn S. Borge-Jensen, 01/02/2022 <i>Oddbjørn Borge-Jensen</i>

D.5 Minutes of Meetings

D.5.1 Client meeting summaries

Meeting Minutes – 14.01.2022

I. In attendance

Yan senko, Nicholas Sellevåg, Oddbjørn Jensen, Fabian kongelf, Erik Hjelmås

II. Discussion

Erik Hjelmås described how he has used Gitlab to publish websites for his subject DCSG1005. It is aesthetically pleasing, especially rendering of code. Unfortunately, Erik received negative feedback from students as they were confused by the buttons on the Gitlab interface. This amongst other reason is why Erik wants to research a dedicated website solution through Gitlab Pages.

Erik Hjelmås is available for meetings every other week, in some periods every three weeks.

Group members were introduced by Erik Hjelmås to the standard setup he has explored of creating automated Gitlab Pages in Gitlab. For example, it is done through CI/CD pipeline and defined in a "Gitlab-ci.yml" file. The pipeline is triggered whenever a commit is pushed to the repository.

Erik Hjelmås elaborated on automation is a key aspect of development for thesis.

What Gitlab instances exists?

There are four usefull gitlab installations at NTNU according to Erik.

What files should be transformed into websites?

As a baseline for thesis, only markdown files are to be transformed into HTML.

Meeting Minutes – 27.01.2022

I. In attendance

Fabian Kongelf, Yan Senko, Oddbjørn Jensen, Nicholas Sellevåg, Erik Hjelmås

II. Discussion

Erik can participate in meetings whenever we want to, though try and keep the meetings on wednesdays.

Survey on how different courses use gitlab/hub pages to create course pages (contact UIO website we got provided?) (Roger Antonsen)

Interview list:

Roger Antonsen (Github Pages user, UIO)

Frank Aleksander Kremer (Design Science teacher, uses Github Pages)

Mariusz Nowostawski (created setup for Databaser)

ivar farup (Professor Institutt for datateknologi og informatikk)

Meeting Minutes – 02.02.2022

I. In attendance

Fabian Kongelf, Yan Senko, Oddbjørn Jensen, Nicholas Sellevåg, Erik Hjelmås

II. Discussion

The developed system should as a default include a home page for a course. Documents given as input will be subsequently linked within the home page.

Documents as input is limited to markdown files. For example, “docx” file extension “does not belong on websites” - Erik Hjelmås.

Meeting Minutes – 28.02.2022

I. In attendance

Yan senko, Nicholas Sellevåg, Oddbjørn Jensen, Fabian kongelf, Erik Hjelmås

II. Discussion

Erik Hjelmås recommended contacting Mariusz nowostawski and ask him for permission to use his setup for access control in Gitlab as a refference.

Mariusz is depentend on script ran with administrator privileges. Script is dependent on all students being enrolled priorly.

Erik Hjelmås elaborated on small errors in user access control could lead to major problems. For example having students gaining access to exam documents.

Erik mentioned Ivar Farup and Rune Hjelsvold have courses using Mariusz access control setup. Perhaps interview them to gain insight into the user experience.

D.5.2 Guidance meeting summaries

Meeting Minutes – 18.01.2022

I. In attendance

Fabian Kongelf, Yan Senko, Oddbjørn Jensen, Nicholas Sellevåg, Guoqiang Li

II. Discussion

Give advice regarding task and research question. Thesis could be a comparative analysis between blackboard and the developed system.

Weekly 30 minutes meetings, as default on Tuesday 15:15-15:45

Gather information of main product features of blackboard and other LMS.

Students and teachers have used blackboard for a long while, we should have Questionnaire for them. Find out what blackboard is lacking in, then make our solution be a good replacement. Design a questionnaire that leans towards something we beforehand. Online website, ten minutes,

Prepare for a project plan within january, send to guidance few days before final delivery. Within easter holiday we should have a draft. Demo in march, April.

Meeting Minutes – 01.02.2022

I. In attendance

Fabian Kongelf, Yan Senko, Oddbjørn Jensen, Nicholas Sellevåg, Guoqiang Li

II. Discussion

Ask the question why we are developing a feature. And document findings in Definition of done, for example a sufficient answer should come from:

- Research paper (ask Erik for the subject name of such papers)
- Experience from blackboard

Future meeting callings should only include the obligatory people to be present as everyone listed gets an email which can be confusing

Meeting Minutes – 22.02.2022

I. In attendance

Fabian Kongelf, Yan Senko, Oddbjørn Jensen, Nicholas Sellevåg, Guoqiang Li

II. Discussion

Group members presented the high level diagrams, in addition the file structure model.

1. Difficult to understand how the content becomes input for the rest of the processing
1. legend/description for different items. for example markdown and jekyll

Yan Senko showcased the Bachelor thesis template:

1. Abstract chapter before table of content
2. methodology - before everything we do, entails the scientific plan, "we will do survey, we will conduct questionnaires, we will design our system and implement our prototype and look at the security, and have a demo to showcase the security".
3. Conclusion should entail the pros and cons to our system.

Meeting Minutes – 15.03.2022

I. In attendance

Fabian Kongelf, Yan Senko, Oddbjørn Jensen, Nicholas Sellevåg, Guoqiang Li

II. Discussion

have intervju with Erik specifically about what he dislikes. then we can create a list with functionalities and improvements we think are possible and if he likes it we can create goals of that list to be implemented in the purpose section.

Perhaps ask for Erik Hjelmås (client) to rewrite bachelor thesis to have focus on user-friendliness and add functionalities missing in blackboard.

Meeting Minutes - 05.04.2022

I. In attendance

Fabian Kongelf, Yan Senko, Oddbjørn Jensen, Nicholas Sellevåg, Guoqiang Li

II. Discussion

Received feedback on models from Li:

HighlvlModel - model

* perhaps add students in the highlevel diagram after curriculum

FileStructure - model

* divide folder into sub models

Meeting Minutes – 19.04.2022

I. In attendance

Fabian Kongelf, Yan Senko, Oddbjørn Jensen, Nicholas Sellevåg, Guoqiang Li

II. Discussion

In report, bring forth negative user experiences from survey. Thereafter, thesis should include the features implemented in developed system to overcome said negative experiences in other LMS such as Blackboard Learn.

Summarize interviews and add to report. in practice, remove unnecessary notes from interview documents.

Reformulate task description into research questions. The questions are vital to adhere to scientific studies.

from a scientific approach we would read research questions. they are essentially the scope of the project. in the end we try to end the questions, we should be able to answer the questions. Do not leave the question open by the end of the report.

D.6 Task

Git da gitt!

Hverken studenter eller ansatte ser ut til å være spesielt glad i eksisterende LMSer (Learning Management Systems som Blackboard og Itslearning). LMSer skal gjerne løse følgende seks hovedutfordringer:

1. Gi lett tilgang til åpen informasjon (emnebeskrivelse, timeplan, undervisningsplan, læringsmaterieell, o.l.) om et emne.
2. Gi lett tilgang til lukket informasjon (løsningsforslag, ikke-offentlig pensum, opphavsrettbegrenset materiale, o.l.).
3. Publisere kunngjøringer (viktig informasjon underveis i en emnegjennomføring)
4. Ta imot innleveringer og gi tilbakemeldinger.
5. Utføre digitale tester.
6. Diskusjonsforum.

Oppdragsgiver ønsker å utforske muligheten for å bruke andre eksisterende systemer vi allerede har i bruk for å løse disse seks utfordringene. Eksempelvis:

1. git
2. git
3. git
4. git
5. Inspira
6. Piazza

Oppgaven

Git(t) at studentene (og lærerne) behersker git, hvor godt kan punkt en til fire løses av git? I første omgang ønskes det at gruppen utforsker GitLab og [GitLab Pages](#) (evn også GitHub og [GitHub Pages](#)) mtp å møte utfordring nummer en og to:

- Hvordan bør en webside med all nødvendig åpen informasjon for et emne se ut fra et studentperspektiv i et IT-emne?
- Hvordan kan en slik webside realiseres med GitLab pages?
- Hvordan kan en emneansvarlig/faglærer ivareta aksesskontroll til de tre kategoriene:
 - a) åpen informasjon
 - b) informasjon kun for deltakerne i emnet
 - c) informasjon kun for emneansvarlig/faglærer (f.eks. fremtidige eksamensoppgaver)
- Er det også mulig å løse utfordring tre og fire med git?

Oppdragsgiver og kontaktperson: NTNU, IIK ved Erik Hjelmås (erik.hjelmås@ntnu.no)

Glossary

Containerization Containerization is a form of operating system virtualization where applications run in isolated user spaces called containers [61]. 13

Control group cgroups (abbreviated from control groups) is a Linux kernel feature that limits, accounts for, and isolates the resource usage (CPU, memory, disk I/O, network, etc.) of a collection of processes [8]. 13

Development & Operations DevOps is a set of practices that combines software development (Dev) and IT operations (Ops). It aims to shorten the systems development life cycle and provide continuous delivery with high software quality [62]. 70

Docker Docker is a set of platform as a service products that use OS-level virtualization to deliver software in packages called containers [63]. 13

Downstream job A downstream job is a configured project that is triggered as part of a execution of pipeline [29]. 62

GitLab Job artifact Jobs can output an archive of files and directories. This output is known as a job artifact [**artifact**]. 71

Gitlab runner GitLab Runner is an application that works with GitLab CI/CD to run jobs in a pipeline. [64]. 60

Hashing Hashing implies the use of an algorithm for creating a "fingerprint" of the provided data. This "fingerprint" is unique to the input: a minor change changes the hash value completely[65]. 58

input cost In encryption, the input cost determines the amount of iterations an algorithm is set to go through [66]. 58

Jira Jira is a proprietary issue tracking product developed by Atlassian that allows bug tracking and agile project management [67]. 8

Layout Layouts are templates that can be used by any page in a site and wrap around page content[68]. 30, 31

Linux kernel The Linux kernel is a mostly free and open-source, monolithic, modular, multi-tasking, Unix-like operating system kernel [11]. 13

Liquid template language Liquid is a template language used to load dynamic content in the pages of online stores. Website designers and developers can use a template language to build webpages that combine static content, which is the same on multiple pages, and dynamic content, which changes from one page to the next[7]. 30

Markdown Markdown is a lightweight markup language that you can use to add formatting elements to plaintext text documents. Created by John Gruber in 2004, Markdown is now one of the world's most popular markup languages[69]. 30

Microsoft Teams Microsoft Teams is the ultimate messaging app for your organization—a workspace for real-time collaboration and communication, meetings, file and app sharing, and even the occasional emoji! All in one place, all in the open, all accessible to everyone [70]. 27, 28

Microsoft Teams Cards A card is a UI container for short or related pieces of information. Cards can have multiple properties and attachments and can include buttons, which trigger card actions. Using cards, you can organize information into groups and give users the opportunity to interact with specific parts of the information [71]. 46

Namespace Namespaces are a feature of the Linux kernel that partitions kernel resources such that one set of processes sees one set of resources and another set of processes sees a different set of resources [72]. 13

Node Packet Manager npm is the world's largest software registry. Open source developers from every continent use npm to share and borrow packages, and many organizations use npm to manage private development as well [73]. 67

Node.js Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine [74]. 67

Open Authorization OAuth is a standard for access delegation, a common way to authenticated users based on their credinsals form other sources. For instance the ability to log on to sites never before visited with a google or facebook account [75]. 63

Operating system An operating system is system software that manages computer hardware, software resources, and provides common services for computer programs [76]. 14

Phishing An adversary attempts to pretend to be a legitimate actor, in order to make the target abide their requests, such as downloading a malicious file or providing sensitive information [77]. 58

Process A running program [78]. 13

Runtime Runtime is a system used primarily in software development to describe the period of time during which a program is running [28]. 48

salt a random set of characters, appended to the user password before being hashed [65]. 58

Software as a Service Software as a Service (or SaaS) is the least flexible service provider alternative from the customer side. The seller provides only the service, while also handling its infrastructure and development platform [79]. 9

Syntactically Awesome Style Sheets Sass is a stylesheet language that's compiled to CSS. It allows you to use variables, nested rules, mixins, functions, and more, all with a fully CSS-compatible syntax. Sass helps keep large stylesheets well-organized and makes it easy to share design within and across projects[80]. 31

Upstream job An upstream job is a configured project that triggers a project as part of its execution [29]. 62

Web element An element is a part of a webpage. In XML and HTML, an element may contain a data item or a chunk of text or an image, or perhaps nothing. A typical element includes an opening tag with some attributes, enclosed text content, and a closing tag[81]. 30

Web interface A Web user interface or Web app allows the user to interact with content or software running on a remote server through a Web browser. The content or Web page is downloaded from the Web server and the user can interact with this content in a Web browser, which acts as a client [82]. 21

Acronyms

CSS Cascading Style Sheets. 30, 31

HTML Hypertext Markup Language. 30, 31

OS operating system. 13–15

Sass Syntactically Awesome Style Sheets. 30

