Technische Universität München

Department of Informatics

Master's Thesis in Informatics

iLab Exchange Platform

Julius Polar

Technische Universität München

Department of Informatics

Master's Thesis in Informatics

iLab Exchange Platform

iLab Exchange Platform

Author	Julius Polar
Supervisor	Prof. Dr.-Ing. Georg Carle
Advisor	Dr. Marc-Oliver Pahl, Stefan Liebald
Date	September 15th 2017

Informatik VIII
Chair for Network Architectures and Services

I confirm that this thesis is my own work and I have documented all sources and material used.

Garching b. München, September 15th 2017

Signature

Abstract

Lab exercises are an essential part of computer science education. This Master's thesis discusses a lab sharing platform where labs, or practical exercises, can be shared between lecturers. The iLab Exchange Platform is designed to facilitate the exchange of labs using a community website. To ensure quality of the labs, feedback is gathered from students, then analyzed and visualized. Integration possibilities with an existing e-learning software called the labsystem are researched and described. The platform uses Git as backend to store course materials. GitLab projects and groups are used with GitLab user roles to control access to the labs and solutions. The GitLab API is used to deliver information from the GitLab repository to the portal website. The described system is implemented, deployed and its usability is evaluated.

Zusammenfassung

Laborübungen sind ein wesentlicher Bestandteil der Informatikausbildung. Diese Master-Arbeit diskutiert eine Lab-Sharing-Plattform, über die praktische Kurse oder Laboraufgaben zwischen Lehrernden geteilt werden können. Die iLab Exchange Plattform ist derart gestaltet, dass sie den Austausch von Laboraufgaben durch das Verwenden einer Community-Website erleichtert. Um die Qualität der Laboraufgaben zu gewährleisten, wird Feedback von den Studenten gesammelt, analysiert und visualisiert. Integrationsmöglichkeiten mit einer bestehenden e-Learning Software, dem Labsystem, werden erforscht und beschrieben. Die Plattform benutzt Git als Backend zum Verwalten von Kursmaterialien. GitLab-Projekte und -Gruppen werden mit GitLab-Benutzerrollen verwendet, um den Zugriff auf die Laboraufgaben und Lösungen zu beschränken. Über die GitLab API werden Informationen aus dem GitLab Repository auf die Portal-Website übertragen. Das beschriebene System wird implementiert, bereitgestellt und seine Nutzbarkeit wird evaluiert.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Education has a large and important role for people around the world. People with higher education have a better chance of receiving better paying jobs, therefore better possibilities to improve their lives. We live in a knowledge society that processes information and knowledge in ways that maximize learning [1]. Traditionally, education system has students and teachers physically interacts in the classroom. Recent advances in technology has resulted in changes in traditional media.

More and more fields are adapting technology, including education. This trend has improved teaching and learning activities in higher learning institutions, especially in developed country [2]. In recent years, due to the growth of the Internet, has changed our view of education [3]. e-learning as a concept emerged around 20 years ago, when institutions started to use technology to deliver learning materials, such as CD based training programs and distance learning using computer networking. Various learning technologies have been developed to integrate technology into traditional learning methods. e-learning mainly takes the form of online courses, with learning management systems organizing and delivering online courses [4].

A popular trend in e-learning are MOOCs (Massive Online Open Courses). Popular MOOC platforms, such as Coursera and edX, are offering their courses to the general public for free. Students around the world can join the courses they are interested in and learn at their own pace. There are plenty of courses available, covering different topics. Some MOOCs offering similar subjects can cover different topics. Choosing the course that covers the topics they are interested in is a commitment for the student, while choosing what topic to cover in a course a challenge for the teacher. Courses offered by different platforms can have intersecting topics covered by another platform.

Instead of creating the course content from scratch, some contents can be reused from another platform. By exchanging course materials already made, a course manager can create a better course by selecting relevant contents. We see that sharing course content can help both the student and teacher. With an exchange platform, materials from other courses can be used for another, saving time to create a modules for similar topic. A course can consist of lectures, quizzes, practical exercises, and materials. In this thesis we define labs as a set of practical exercises.

The Chair of Network Architectures and Services of Technical University of Munich has developed an e-learning tool to support courses, the *labsystem*. This *labsystem* is based on *iLab* concept. The iLab is a didactical concept that delivers course in three parts: lecture, pre-lab, and lab exercises. The lecture is given every week to introduce the necessary materials and knowledge for the week's lab exercise. Then a pre-lab is given and must be completed to unlock the lab exercise. The pre-lab consists of brief introduction to the materials and multiple choice questions, to give a feel of what the lab would look like. The pre-lab and lab exercise is delivered using the *labsystem*. In the Technical University of Munich, it is available in two courses, *iLab* and *iLab2*. In *iLab2*, students create their own lab exercise and uploads it to a pool of lab exercises. This creates a platform of sharing course materials.

This concept can be extended to other courses, offering a platform of course exchange. The goal of this thesis is to construct a community website that presents current labs (learning modules) as an exchange platform. An important module to be developed for this platform is gathering and visualizing lab feedback such as tracked times, or qualitative feedback. Our implementation goal is to make a lab exchange, but the same concept applies to courses too. Throughout this thesis, we analyze a course exchange, and design and implement a lab exchange specifically for the *labsystem*.

1.1 Goals of the thesis

The goal of this thesis is to construct a community website that presents current labs (learning modules) as an exchange platform. An important module to be developed for this platform is gathering and visualizing lab feedback such as lab difficulty and satisfaction, which is used to help instructors to improve their labs.

1.2 Outline

The thesis is structured as follows. In Chapter 2 e-learning and its components are briefly described. Then the current *labsystem* and *iLab* concept and how it can be expanded to a course exchange platform are described. We describe the exchange platform, and elements necessary to construct one. Then available frameworks are discussed to assess its advantages and disadvantages for the exchange platform.

In Chapter 3 platforms with e-learning and sharing components - MOOC, YouTube, Facebook, and Moodle are examined. We consider how they offer solutions for our requirements and review them.

In Chapter 4 we describe each requirement in more detail, discussing how to create the exchange platform based on what we have learned so far. We describe the current *labsystem* workflow and what features it already offers. Possible workflow from other platform, most notably Moodle, are discussed. We then describe how Git can be used to construct the backend to store contents for the exchange platform, and how a frontend portal can be constructed based on such Git architecture.

In Chapter 5 the implementation of the exchange platform are described, including the workflow of the exchange platform, how to setup and implement the GitLab backend and frontend portal, and how to integrate them with the current *labsystem*.

In Chapter 6 the exchange platform is evaluated by comparing its features with similar works. Features related to feedback and lab sharing is compared, and their advantages or disadvantages discussed.

In Chapter 7 we conclude by summarizing the exchange platform concept and its implementation. Possible future development for this works is presented.

1.3 Methodology

The thesis is conducted as follows. Since the exchange platform is an e-learning platform, we discuss e-learning and its components. After we breakdown e-learning components, we discuss which parts are relevant for the exchange platform. We then discuss the current state of *labsystem* and *iLab* workflow. Based on those, we describe the requirements of the exchange platform.

Similar e-learning platforms are discussed and compared. We compare their features and how they provide solutions for the exchange platform. Having defined the necessary requirements, we then describe the required components in more detail, and how they can be constructed.

A working implementation of the described system is developed. The prototype system is hosted in a virtual machine during development, and made available for public access. The prototype environment is isolated from the currently running system, but integration with currently running system is described and documented.

Chapter 2

Analysis

To give context for describing the problem domain, we briefly discuss e-Learning. The components of e-learning are listed, and the necessary ones identified and discussed. Then the current state of the iLab system is described. Next, a course exchange platform is described. Finally, based on current system and requirements, we defined research questions and propose the requirements of the exchange platform.

2.1 e-Learning Workflows

The *labsystem* described in this thesis is an e-learning platform, built to support web-based learning in courses. In order to get an understanding of what components are needed for the exchange platform, e-learning workflow is briefly discussed. We break down necessary components for creating an e-learning platform and discuss components necessary for the exchange platform.

An e-learning platform has many workflows [5]. Here only components relevant to the thesis are discussed. The components listed are the possible building blocks to build the exchange platform with. Those components are:

- Course information and syllabus (Section 2.1.1)
- Content Catalog (Section 2.1.3)
- Material (Section 2.1.2)
- Communication (Section 2.1.4)
- Feedback (Section 2.1.5)
- Learning analytics (Section 2.1.6)

2.1.1 Course Information

Since the exchange platform hosts many courses, it is important to show the information of the course to the users. The course information is the first information shown when a student or a teacher search or browse through the course list.

The main page or landing page of a course offers general information of the course. In the page, the student can expect to find information such as [6]:

- Course syllabus
- Prerequisite courses (if needed)
- Grades or scoring information
- Credits given upon completion
- Course schedule
- Course registration (if needed)
- Payment or cost (if needed)
- Reviews and comments

In the exchange platform, the course information page should inform the student about what topics are covered in the course, whether the topics interest them, time commitments, and general impression from previous students.

2.1.2 Course Material

Since the platform is meant to share course material, it is relevant to know how the course materials look like. In this section we discuss how the materials can be presented. These materials are shared in the platform, so a format is needed to store the materials.

Course materials can be viewed directly from the course website without using non-web based media, like reading an article in the web. Multiple medias such as images or videos can be added to augment the materials to improve learning [7].

The course materials are uploaded online for the students. They can be in several forms:

- text: pdf documents, presentation slides
- video: lecture recordings
- audio: lecture recordings (audio only)
- interactive form: embedded questions

The materials can be delivered via multimedia to help with the learning experience. YouTube, for example, hosts plenty of recorded lectures and educational materials, but only delivers them in video format. Most Massively Open Online Courses (MOOC) providers delivers their materials in multiple medias (Section 3.1).

Additional materials can also be added, such as extra readings and external links to references. For topics that refers to another course, here is an opportunity to promote the external course for further interests.

The embedded medias in the course materials (videos, images) can be stored in the platform's content management system (and the exported course links to it), or included in the exported format, or hosted in another platform like YouTube for videos.

The shared contents need to be saved in a digital format. Moodle, for example, offers an export feature for quizzes in xml format. The xml file contains the quiz metadata and content. It can also be stored as rows in a database and exported as an sql file.

These materials can then be shared into the exchange platform. **The exchange platform requires an uploading feature to upload these materials in order to share them[R01].**

2.1.3 Content Catalog

Once stored in a format, the material can then be shared in a public list of resources. The exchange platform need to store this material, and let the users locate the resource they want. Now we discuss how to store the shared materials, how to present the list of materials to the user, and how to help them find materials they want.

The shared contents can be stored in multiple ways:

- Content management system. The shared contents are stored in a CMS that also hosted the exchange platform.
- Database entries. The content can be broken down into database column entries, and the whole shared item as a row in the database. It can also be stored as a binary file in the database.
- Public repository. This method uses a storage provider like Google Drive or Dropbox to store the exported course materials. A public SVN or Git repository can be used to provide versioning as well.

These shared contents need to be listed for interested users to browse through [R04]. For example, Moodle.net [8] provides a list of community uploaded courses and exercises. Here, a brief explanation of the item is shown, along with the metadata of the course, such as the creator, subject, or language. In some items, a link is provided to join the course on the university's Moodle.

To make finding contents easier, a search or grouping feature are usually offered. **A search feature is crucial to help users find the content they want [R05]**. Moodle.net provides a search text field and courses that can be joined or downloaded. MOOCs like Coursera or EdX group their courses by either universities or subject. Coursera also provides a set of related course called specializations, and the list can be filtered as such.

Other platforms, not necessarily educational, also hosts contents and provides a way to find specific items. YouTube for example, provides a search text field, a trending page, and group by category. It also provides recommendations based on watch history. Some platform provides tags to find similar content, such as Imgur (image tags) and Steam (genre tags).

2.1.4 Communication

A communication platform, student to student and student to lecturer, is essential to any e-learning platform. Great communication channel can help content creators to improve their creation. However, developing these is out of the scope of the exchange platform, since each courses usually offers their own communication platforms, making it redundant. There is also a problem to encourage students to use these in order to generate discussions. To help improving created content, a feedback system is discussed in the next section.

2.1.5 Feedback

One of the main focus of the exchange platform is gathering and analyzing feedback. The exchange platform requires feedback for content sharers to improve their materials. **Feedback needs to be gathered, and then analyzed to improve shared materials [R07].**

Constructing meaningful content requires meaningful feedback, in this context, from the students. Feedback using online formats can provide more and lengthier open-ended comments, with more qualitative detail than is likely to be found in traditional evaluations [9].

Feedbacks can be personally or anonymously gathered, qualitatively or quantitatively. Results from quantitative feedback can be aggregated and processed to provide a numerical score, which can be used to compare an item against another. Qualitative feedback is used to gather comments or opinions that cannot be represented in numbers. These two are often used alongside each other, for example, a star rating with a short review.

Feedback can be gathered with many methods [10]. Here we only discuss the feedback gathering via the Internet, since the exchange platform is built as a website. The feedback methods considered are:

- Survey or questionnaires
- Email questionnaires or forms
- Customer reviews or comments
- Ratings
- Social media

Below, these methods of gathering feedback and their usage in the exchange platform are described in more detail.

2.1.5.1 Survey

A survey is brief, voluntary, and can be asked after learning activities. Surveys can ask for students' satisfaction with the shared materials. Since it can be done quickly, survey can be used to quickly gather inputs after finishing activities. Online survey have several advantages [11], such as access to unique population, and saving time and money. However, it also suffers from sample problems, incomplete or unacceptable responses, and multiple submissions [12].

Survey can give teacher a general idea of their audience regarding their knowledge of the topic and general satisfaction of the course. Some metrics from the survey are also quantitative, average satisfaction for example, and can be used for statistical analysis of the course.

A survey assessing satisfaction on a course can ask the following questions:

- general satisfaction: are you satisfied with the course? did you learn something new? how much time did you spent on this course?
- materials: are given materials sufficient? how are the quality of the materials?
- difficulty: do you think the material is easy/hard?
- general rating: can be a 10 or 5 star rating, like/dislike, review

The materials can then be revised based on the result of the survey. For example, if the survey comments on a lack of medias, the teacher can include more informative images or videos. Or if a lot of students English level is basic, and the materials is written in a more advanced english, an abridged or simplified version can be supplied.

Some results of the survey can be displayed in the exchange platform. For example, course difficulty survey result can be useful for potential students.

2.1.5.2 Email questionnaires

Email questionnaires are a form of survey conducted by sending email containing survey forms to recipients. Compared to survey mentioned last section, which is conducted in the platform itself. A study found that a Web survey application achieved a comparable response rate to a mail questionnaire when both were preceded by an advance mail notification [13]. Another motivation to use email surveys is that it can be done outside a time constraint, for example a course survey might be closed after the end of that course. However, the survey mail might get lost in the recipient's mailbox or spam filter.

2.1.5.3 Reviews

Reviews constitute evaluations by customer of the product or service on a website, and are displayed next to the product description in order to enhance perception and improve the perceived communication characteristics of the medium [14].

For the exchange platform, reviews are important to potential new students, telling them experiences of previous students, overall satisfaction, and help them decide whether to take the course. It is also helpful for the course creator to have a qualitative feedback on their course. Written reviews can reveal more personal experience relevant to the potential student. They can point out good or bad aspects they experience, such as whether the course is time consuming, the instructions are clear and helpful, etc.

The accumulated student rating and review can be displayed in the course page to provide future student a brief overview of the course, from fellow students. In several platforms, reviews are rateable to help filter lazy reviews and promote accurate and well written reviews, with the best rated and latest of the reviews displayed on top of others to make it most visible. Encouraging users to rate reviews can have a positive effect on controlling the overall number of resources in the system [15].

2.1.5.4 Ratings

Ratings are a score given to a content; in this case, a course, lab exercise, or its material. A rating, such as like/dislike (YouTube, Facebook) or five star rating (Coursera) serves as a quantitative feedback that can be averaged and displayed to show the average consensus or satisfaction from previous students.

The exchange platform can show a numerical score to shared materials to provide a quick overview to interested users, compared to reading comments or reviews. Ratings are often used in conjunction with written reviews, with the reviews justifying or explaining the given score.

Different platforms uses different systems for user ratings. Youtube uses a like/dislike system to their videos. Similarly, Steam uses recommended/not recommended for their user reviews. A five star rating is also commonly used, for example, by Coursera. The like/dislike method offers a much simpler rating, while lacking a moderate opinion offered by five star system.

Rating system can also be used to rate reviews, similar to Facebook's liking a reply to a post. This allows users to mark or approve reviews that they agree with.

2.1.5.5 Social Media

With the rising popularity of social media, it has become a platform to gather feedback. Social media platforms provide tools for the exchange platform to ask and gather feedback from its users. For example, a Facebook fan page can be made as a hub for its users to share comments and experiences. Hashtags can be used to filter feeds or posts related to their experience.

Using social media to gather feedback has advantages, for example, the users might already have social media accounts and use them regularly. Some feedback tools listed before are also already implemented as a part of the social media platform, such as comments, ratings, and surveys. However, although the infrastructure to support social media already exists in most universities, instructors have been slow to adopt the tool as an educational one [16].

2.1.6 Learning Analytics

So far, the feedback gathered is voluntary, based on student's experience. This feedback is meant to help curate contents of the exchange platform. **Now we want to gather data from the student's usage of the *labsystem* to help with learning, using learning analytics [R08].**

Learning analytics is the measurement, collection, analysis, and reporting of data about learners and their contexts, for purposes of understanding and optimising learning and the environments in which it occurs. Learning content can be delivered in modular packages, providing learners with materials relevant to his or her analyzed profile, goals, and knowledge [17].

An example of learning analytics usage in e-learning is a recommendation system. The e-learning system gathers data about the learner and creates a profile, then suggest courses or lessons based on the student's interests or recently taken courses. Another use of learning analytics for teachers is to see how the students performs on courses. For example, time to solve questions can be tracked to gauge difficulty of the questions.

The gathered data is then visualized to help teachers understand the data. For students and teachers, it can be useful to have a visual overview of their activities and how they relate to those of their peers or other actors in the learning experience [18]. **Analyzing and visualizing usage data is important to provide better understanding about the data [R05].**

Gathering feedback via learning analytics is not a focus of this thesis, but its implementation possibilities are examined further in Section 4.1.8. The *labsystem* has some learning analytics function that should be considered. A fellow colleague writes a thesis regarding improving e-learning through statistical feedback [19]. She noted time needed on exercises, worst answered questions, and emotion feedback as artifacts to gather.

2.2 iLab

In the previous section, e-learning and its workflows are described. Before we describe the exchange platform, we must first discuss its foundation, the iLab.

iLab (Internet lab) is a lab course where students work in an controlled environment and solve networking related exercises. An example lab session would be a firewall exercise, where students connect the proper cabling to different computers, configure and test firewall rules, and answer firewall related questions. It is offered in 2 courses, iLab and iLab2.

iLab is done in a team of two students. The materials are delivered in a weekly basis, one lab topic in a given week. A lab is made of two parts, a prelab and a lab. The prelab provides a brief overview and reading materials for the week's topic. In the lab session, students perform tasks and experiments on a specially made laboratory. The students then answer the given questions. An optional feedback form is presented after the end of each lab.

iLab2 offers the same lab structure, but students can choose lab modules made by previous iLab2 students. After finishing with theirs, they make a lab module of their own, and that lab module is submitted to the pool of available labs. These lab modules are curated based on student's feedbacks.

The *labsystem* used by iLab has components necessary to serve as a basis of an e-learning platform: module creation features, basic feedback form, course evaluation, and module reviews.

The concept of sharing premade lab exercises is the idea of the exchange platform, described in the next section.

2.3 Exchange Platform

The exchange platform is a platform where users can share educational contents. Here we analyze possible aspects of the exchange platform.

The exchange platform is about sharing educational contents. Different courses have overlapping topic of interests, and share common materials, for example, a formula sheet used by Mathematics and Physics students. By sharing materials, similar courses or exercises can be constructed faster by using the shared contents. Students can also find interesting contents shared in the platform.

First of all, what does the exchange platform shares? Educational contents are varied, and there are benefits in sharing larger or smaller contents at a time, depending on the needs. Here are the possible contents that can be shared in the platform:

- Course media. For example, a document file, an image, or a binary file. Since this is the most basic building block of a content, this is the smallest unit that can be shared. Sometimes a file need to be shared without sharing the whole course or exercise, for example, an annotated pdf, a documentation file, or a presentation file.
- Lectures. Can be composed of multiple media like text, images, or videos.
- Exercises or lab modules. This is based on the concept from course iLab2 mentioned earlier. Lab exercises can be shared to other courses that shares a similar topic.
- Quizzes. Similar to previous point. By sharing quizzes, it provides qualified set of questions to evaluations of the same topic.
- Courses. Share the entire course, including its materials, lectures, quizzes, and exercises.
- Tools. Share tools used by a course or lab, for example a script or executable file.

The format of these shared contents also need to be defined. The *labsystem* provides a lab creation tool to let students create their lab module, where the lab exercise is defined by the *labsystem*. The *labsystem* then shows the contents of the lab exercise, along with instructions for the students to follow. The exchange platform needs to define the format and procedures of creating a course or lab exercise. For example, the shared lab exercise can use the iLab's *labsystem* similar to iLab2. Another way is to submit the instructions and materials without using the *labsystem*.

Since there are materials being shared, the next point is, who shares the contents? As a part of the syllabus, students of iLab2 create their own lab modules and contribute them to the available lab modules for the next set of students. However in the exchange platform, who can share the contents must be defined.

There are multiple possibilities to do this. **A Role Based Access-Control (RBAC) can be used to restrict access to the sharing feature and access to the shared materials[R02]**. The attending students can share materials like iLab2 students. Material sharing can also be restricted to teachers only. This ensures materials are created by qualified personnel. Further roles can also be created, where a role can only create material, but not share it to the platform, and another role to review the material, and then sharing it. This allows a clearer reviewing process before the material is shared to the platform. Access to solutions must also be considered. Some online exercises in MOOCs also offers access for the solutions for its participants, but restricted access should be handled as well.

Another goal of this thesis is feedback. Feedback is gathered and used to curate and review materials so they are up to standards and ensure their quality. For example, a lab module needs to make sure that the instructions actually work. Possible methods of gathering feedback are discussed in Section 2.1.5. **These feedback can be displayed with the materials to show opinion regarding that material [R06]**.

Another point to consider in gathering feedback is whose feedback is to be gathered. For example, everybody that uses the platform can give comments on every shared materials, regardless of whether they have experience that course or lab module. This can be useful in the early phases of the exchange platform, where every user is new, and the former students of the course (before it is shared in the platform) can give their feedback. However, this can also be misused to give irrelevant reviews of the contents. The platform can also allows only students who is taking or complete the course to give their reviews. Another way review material is to only let teachers or instructors to review the content.

After being given feedback, the materials can be revised. **The changes made to these materials need to be tracked via a versioning system [R03]**. A manual log can be created for each changes, but it can be cumbersome to manage. A versioning system can be used instead. An advantage to keeping track of the version is older version of the material can be accessed as necessary. Versioning tools also allows forking, where an item can be forked or duplicated and modified separately like a shared template to quickly branch out a material. The content creator can also learn from looking at the change history to identify mistakes. Versioning provides a powerful way to track and compare versions, retrace errors and explore new approaches in a structured manner [20].

The exchange platform should also consider which shared contents to be versioned. Courses, lab exercises, lectures can be versioned to keep track of their changes, and can be forked to start another courses or lab exercises. The media contents can also be versioned, but it is not necessary, since the feedback is gathered from courses and lab exercises, not their composing medias.

So far we have discussed the possible building blocks for the exchange platform and possible aspects to build it. The exchange platform is about sharing educational material, and curate those materials with gathered feedback. These two points, sharing contents and curating contents are important goals of this thesis.

2.4 Existing Frameworks

Now that we have a description of the exchange platform, we look at other frameworks and look for possible building blocks. Some of them implements some requirements of the exchange platform.

Here are a list of similar frameworks, offering solutions to the exchange platform. Not all of the features are necessary for the exchange platform. These platforms are chosen due to their partial solutions to exchange platform's requirement.

2.4.1 Moodle

Moodle, *Modular Object-Oriented Dynamic Learning Environment*, is a course management system for online learning. Moodle has become a term of its own synonymous with a software package designed to help educators create quality online instruction [21]. Moodle provides framework for some necessary feature for the exchange platform.

Moodle offers a variety of content creation tool. The lesson module allows creation of lessons embedded with multimedia, attached with assessments. Moodle keeps track of student activity to inform the teacher when the students complete their assignments or quizzes. The teacher can also set timeframe and deadline for each content and restrict access afterwards. In whole, Moodle provides an excellent array of tool for learning management.

Moodle saves created question from quizzes and lessons into question bank. The questions in the bank can be reused for another quizzes and lessons. The quizzes and questions can be exported to a file, which then can be imported. However, the question database is only available for the teachers. Whole courses can also be exported using the backup feature. The advantage of this system is that quizzes, questions, and courses are exportable and importable. The disadvantage is that students cannot freely browse the question bank, and they are not rateable. The backup feature also saves many aspects of the course, that are not exchanged in the exchange platform, such as enrolled students, forum posts, course progress, and calendar events.

Some of this course backups can be found in Moodle.net, a place where Moodle users share their courses. Various contents are available ranging from database template, books, and glossary entries. The site must be registered in moodle.org for the course in it to be publishable to moodle.net. The course should also satisfy moodle.net's course approval criteria [22]:

- The course must contain at least one useful, interactive activity
- The course must not contain any illegal, insecure, or copyrighted content
- If the course is for download, the course backup file should be be less than 250MB
- Visitors must be able to actually enrol in the course, with no enrolment key

The advantage of this system is that there is a clear guideline for content submission. The course needs to be approved before appearing in the content list in moodle.net. The disadvantage is there is no user rating without installing an external plugin.

In the Moodle ecosystem, each institution or university can host their own Moodle server, then export their courses and share it in Moodle.net to the public. These shared courses can then be imported and run on other Moodle installation. A more detailed discussion on Moodle can be found in Chapter 3

2.4.2 Google Analytics

As discussed in section 2.1.6, usage statistics are useful for analyzing course performance. To identify what metrics to gather and analyze, two web analytics provider: Google Analytics and Piwik are examined. Here the metrics are listed and discussed, to find which are useful for the exchange platform.

Google Analytics is a web analytics service offered by Google that tracks and reports website traffic. According to Web Analytics Association, "Web Analytics is the measurement, collection, analysis and reporting of internet data for the purposes of understanding and optimizing Web usage." Web Analytics Association also defines metrics for web analytics definitions:

It is important to note that Google Analytics is a tool for web analytics, meant to improve the usage of websites [23]. However, some of its collected metrics are still relevant for the exchange platform, noted below.

2.4.3 Piwik

Unlike Google Analytics, Piwik is an open-source web analytics platform. Due to its open-source nature, the user has complete control of its code, the server it runs on, and the privacy settings of its tracking [24]. It tracks keywords, top page URLs, page titles, user countries, internet provider, operating system, browser, screen resolution, time on site, pages per visit, returning visit, and many more.

Among Google Analytics and Piwik's metrics, here are the ones that can be used by the exchange platform:

- Average visit duration: to track average time spent on each pages on lab modules. It can then be summed up to provide an estimate of time spent on a course.
- Pageviews: along with average visit duration, it can be used to estimate average time spent on each pages on lab modules.
- Visits over time: to estimate how many sessions are needed to finish a lab module.

Both Google Analytics and Piwik also present some of their analysis as charts or graphs, in order to help visualize the analyzed data. For example, a line or bar chart that visualizes the growth of visitors of the website, or a pie chart showing the ratio of mobile devices. In the exchange platform, these data can be visualized as charts or graphs as needed.

2.4.4 Git

Git is a version control system (VCS) to track changes in computer files and coordinate work for multiple people on those files. It is primarily used to manage source code in software development, but can also be used to keep track of changes to any set of files. Every Git repository on every computer is a full-fledged repository with complete history, full version tracking, and independent of network access or a central server [25]. With disciplined use of Git, individual scientists and labs can ensure that the entire timeline of events that occur over the development of a research project are securely logged in a system that provides security against data loss and encourages risk-free exploration of new ideas and approaches [20]. By creating branches, small incremental changes can be developed independent of the primary "master" branch, encouraging experimentation [26].

Websites such as GitHub and GitLab hosted open-source project controlled by Git, enabling people from around the world to contribute to the project. GitHub is the single largest host for Git repositories, and is the central point of collaboration for millions of developers and projects. A large percentage of all Git repositories are hosted on GitHub, and many open-source projects use it for Git hosting, issue tracking, code review, and other things [25]. GitLab is a web-based Git repository manager with wiki and issue tracking features, using an open source license, developed by GitLab Inc [27]. It is offered in Community Edition (CE) and Enterprise Edition (EE).

In the exchange platform, Git can be used to keep track of versions of lab modules. Each courses or labs can be arranged in either separate projects or branches or directories. Changes made to the lab contents can be viewed via Git commit history. Contents can be reverted to previous versions via rollback. Git repositories like GitHub and GitLab also comes with a user access control to manage user privileges, which can be used to keep unauthorized staffs to make changes to the repository. RESTful APIs are also offered by GitHub [28] and GitLab [29] to access their functions without using their web interfaces. This allows a creation of a customizable application or frontend that access these Git repositories.

2.5 Research Problems and Requirements

In the previous sections, we described the exchange platform and its components, and how similar features are implemented in other platforms.

Given the description of the exchange platform, the main research question of this thesis is *How do we create an exchange platform to share and distribute course contents?* So far components necessary to build such platform are presented. Given the description of those components, more questions can be raised: *How do we ensure quality of the course contents? How do we collect input and feedback of the course from students? How do we help users find related course materials?*

In order to construct such exchange platform, it must have several qualities. These requirements are grouped based their purpose: building the exchange platform, presenting contents, and analysis.

The exchange platform should have the following requirements:

R01 Upload materials for sharing (see Section 2.1.2)

R02 Role based access control for sharing materials (see Section 2.3)

R03 Materials versioning (see Section 2.3)

In order to effectively present its content, the requirements are:

R04 List available materials (see Section 2.1.3)

R05 Search shared materials (see Section 2.1.3)

R06 Display processed feedback (see Section 2.3)

And finally, to improve its contents, the analysis related requirements are:

R07 Gather and analyze feedback about the materials (see Section 2.1.5)

R08 Analyze and visualize usage data of the labs (see Section 2.1.6)

In the next chapter, we discuss similar platforms, and examine how they implement solutions for these problems, and how they can be used as blueprints to construct the exchange platform.

Chapter 3

Related Works

In this chapter, several related platforms is discussed. These platforms offers some solutions to the requirements of the exchange platform. Not all of these listed platforms are educational or e-learning platforms. A brief overview of these platforms is discussed, then their implementations of the requirements of the exchange platform is discussed to look for possible blueprints to build the exchange platform.

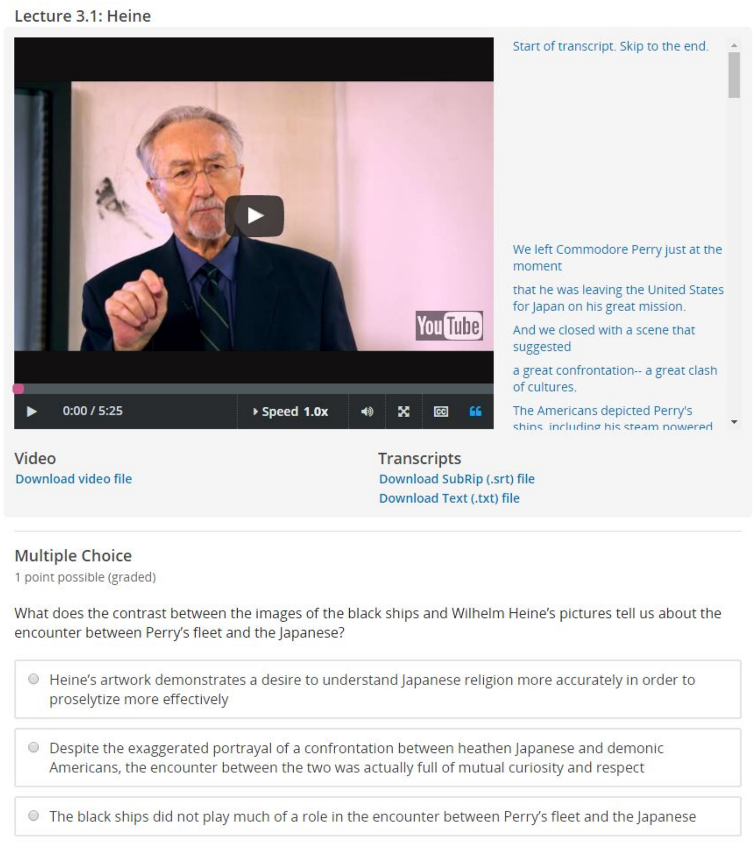
3.1 MOOC Providers

MOOC, Massively Open Online Course, is an online course aimed at unlimited participation and open access via the web. Today, MOOC is the main form of online courses [4]. MOOC also has a problem with high student dropout, with the often cited 90% rate [30]. A study review on massive e-learning design, delivery, and assessment [31] suggest agents to improve and personalize management, delivery, and assessment of MOOC. Such agents helps deliver content with content customization and tutor assistance, assessment based on educational level, and learning analytics.

There are many MOOC providers available in the market offering different courses, with slightly different methods. A more general comparison of Coursera, edX, and Udacity can be found in this survey [32]. Here only implementations of exchange platform features are discussed.

Different MOOC providers present their course material differently. Different courses also present their content differently, for example, some course split a topic into multiple pages containing a presentation video each, while other displays multiple videos in one page. The current iLab system presents materials split into subtopics, with medias as necessary.

Coursera delivers its content mostly via lecture videos. These videos are typically 5-10 minutes long, divided by subtopics. Clicking on these videos take students to a separate page showing only the video. Coursera also provides a pop quiz occasionally during each videos.


Lecture 3.1: Heine

Start of transcript. Skip to the end.

We left Commodore Perry just at the moment

that he was leaving the United States for Japan on his great mission.

And we closed with a scene that suggested

a great confrontation-- a great clash of cultures.

The Americans depicted Perry's ships, including his steam powered

Video
Download video file

Transcripts
Download SubRip (.srt) file
Download Text (.txt) file

Multiple Choice
1 point possible (graded)

What does the contrast between the images of the black ships and Wilhelm Heine's pictures tell us about the encounter between Perry's fleet and the Japanese?

- Heine's artwork demonstrates a desire to understand Japanese religion more accurately in order to proselytize more effectively
- Despite the exaggerated portrayal of a confrontation between heathen Japanese and demonic Americans, the encounter between the two was actually full of mutual curiosity and respect
- The black ships did not play much of a role in the encounter between Perry's fleet and the Japanese

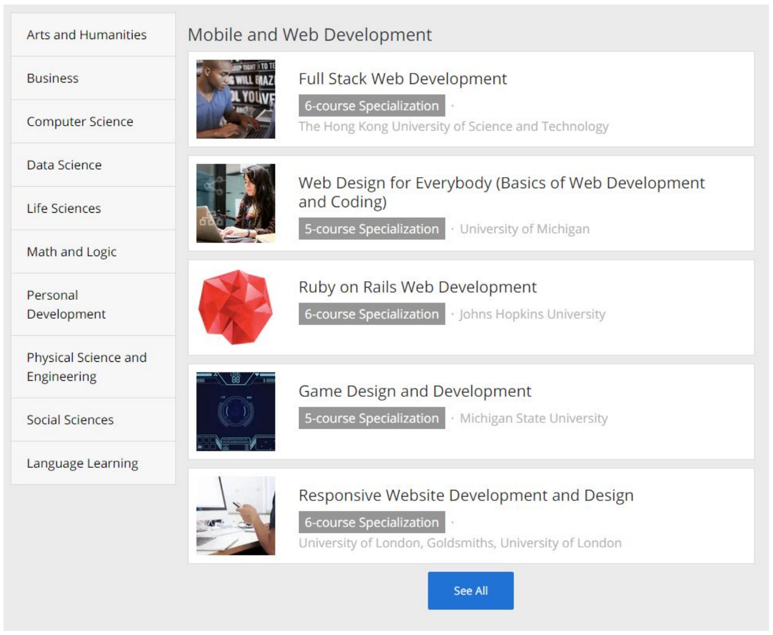
Figure 3.1: edX's course contents

EdX also divides its topic by subtopics. Unlike Coursera, edX shows their course material directly from the course site. EdX also utilizes questions directly from the course page. After answering the questions, answers from other students are shown to compare each other's answer and promote discussion.

The contents delivered by MOOC platforms are not exchangeable, a new course with a similar topic must create their content from scratch.

Coursera, edX, Stanford and other MOOCs combined offers plenty of available courses. The courses are arranged by topics, with other available sorting options. There is also a recommendation system that suggests similar course based on taken course and specialization. The recommendation system is great to look for similar course, or to pursue further topics after finishing a course.

MOOC platforms uses several method to gain course feedback, mainly via course survey and review.



Mobile and Web Development

Full Stack Web Development
6-course Specialization ·
The Hong Kong University of Science and Technology

Web Design for Everybody (Basics of Web Development and Coding)
5-course Specialization · University of Michigan

Ruby on Rails Web Development
6-course Specialization · Johns Hopkins University

Game Design and Development
5-course Specialization · Michigan State University

Responsive Website Development and Design
6-course Specialization ·
University of London, Goldsmiths, University of London

Figure 3.2: Browsing Computer Science courses in Coursera

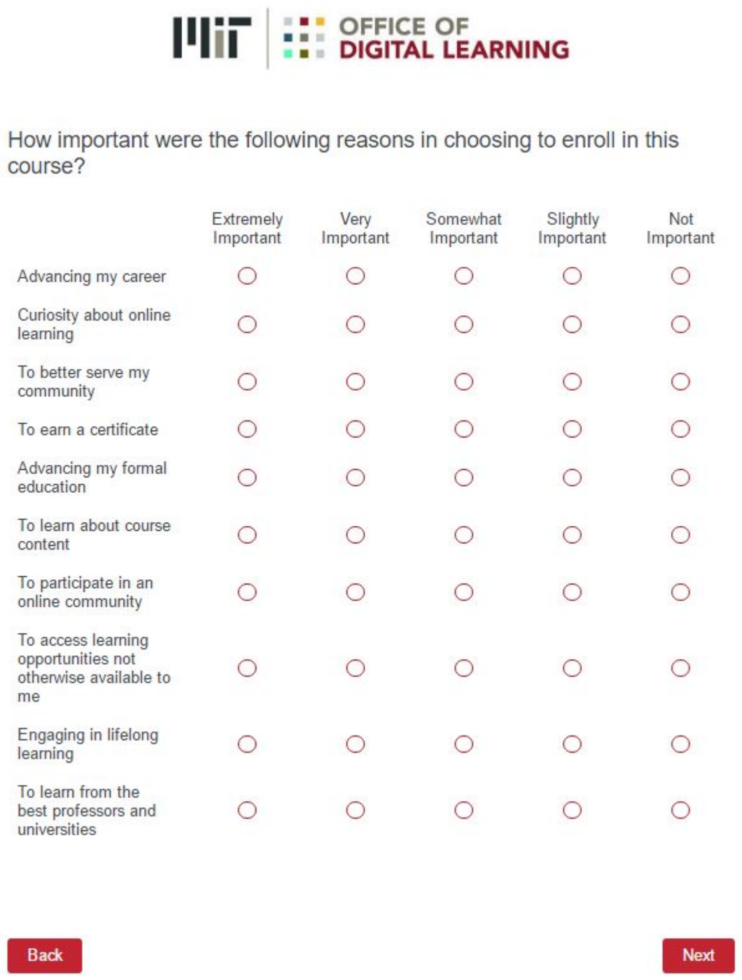
MOOC platforms offers optional survey before registering to the course and after finishing the course, but does not have per-module survey or feedback. The surveys are brief, and can be completed within five minutes. The user reviews are not rateable by fellow students. The list of reviews can also be filtered by everybody, or just course completers.

Surveys at the beginning of the course generally ask for student's motivation, background knowledge, and prior experience. The survey can also ask for the student's predicted course engagement, such as predicted time spent on course, whether they will do the assignments, or whether they intend to complete the course.

Surveys at the end of the course generally asks for student satisfaction, lessons learned, and time spent. The survey can also ask a more specific question about the course material, such as which part is the most difficult, which one interests the student the most, and whether the materials are interesting.

MOOCs also gathers information about its user, some anonymous while others are personal [33] [34]. MOOCs use these information to analyze its user base, behaviour on the system, improving its services, and advertisements. Some information collected are personal, such as username, email address, and birthdate. These are mostly used to communicate with the user, for example, email notifications, assignment deadlines, promotions, and updates. They are also used to keep track of student progress throughout the course.

In summary, here are the requirements of the exchange platform (Section 2.5) and how



MIT OFFICE OF DIGITAL LEARNING

How important were the following reasons in choosing to enroll in this course?

	Extremely Important	Very Important	Somewhat Important	Slightly Important	Not Important
Advancing my career	○	○	○	○	○
Curiosity about online learning	○	○	○	○	○
To better serve my community	○	○	○	○	○
To earn a certificate	○	○	○	○	○
Advancing my formal education	○	○	○	○	○
To learn about course content	○	○	○	○	○
To participate in an online community	○	○	○	○	○
To access learning opportunities not otherwise available to me	○	○	○	○	○
Engaging in lifelong learning	○	○	○	○	○
To learn from the best professors and universities	○	○	○	○	○

Figure 3.3: edX's pre-course survey

these MOOC platforms implement them.

R01 The materials in MOOC platforms are not shared, and a student cannot upload their materials.

R02 Roles are divided to teachers and students.

R03 We could not find any information on this, and assumed there is no material versioning.

R04 MOOC platforms shows their available courses via a course catalog, and group them by universities, fields or specializations.

R05 A search feature is available to search through the courses.

R06 Recent reviews are shown in the course homepage, along with a five-star rating.

R07 Feedback is gathered via survey (pre and post course), written review after finishing a course and a five-star rating.

R08 Data is gathered anonymously and concerning the use of the system, for the purpose of improving its services. The analyzed usage data is not shown to the users.

3.2 YouTube

YouTube, in e-learning, is used mostly to host videos, which is then embedded into the material page. Some university offers full recording of its courses, such as Yale University [35]. However, finding these videos can be difficult. YouTube's channel browsing does not have a dedicated education channel. YouTube's search can be used to find educational videos, but results may vary depends on the topic. The search also prioritizes popular videos, so videos by not very popular uploader or topic might not get any highlight.

YouTube offers feedback in user comments and like/dislike system. Comments can be posted after logging in to YouTube using a Google account. YouTube shows the video's total view and its total like and dislike as a measure of the video's popularity. The like/dislike and comments on the video can be disabled by the owner of the video.

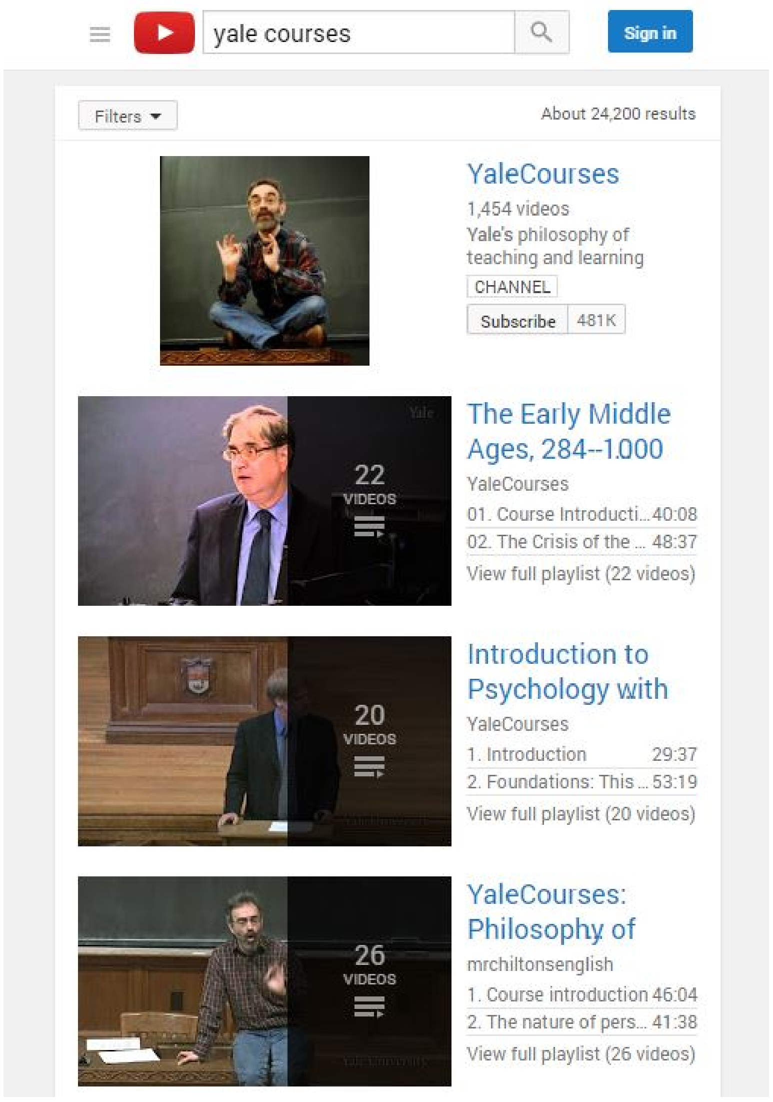
The user comments can be liked or disliked to show approval of the comment. The comments can also be further commented, creating a top level comment on the video, and a second level comment of those top level comments. Only top level comments are comment-able. Both top level and second level comments are rateable by the like or dislike button. By default, YouTube sorts top level comment with the most second level comments, regardless of like/dislike ratio. This is meant to show the most discussed comments or opinion on the video.

According to Google's Privacy Policy [36], Google collects the information the user give when using Google's services, and information the user gave. Google uses the information "to provide, maintain, protect and improve them, to develop new ones, and to protect Google and our users". Google also provides a dashboard where its users can adjust privacy related settings.

In summary, here are the requirements of the exchange platform (Section 2.5) and how YouTube implement them.

R01 Users can upload videos and share them.

R02 Each registered user can post their videos without restriction. A user can assign a channel moderator to moderate comments on the channel's videos. The moderator can only flag comments for removal, but not uploading new videos.



About 24,200 results

YaleCourses
1,454 videos
Yale's philosophy of teaching and learning
CHANNEL
Subscribe 481K

The Early Middle Ages, 284--1000
YaleCourses
01. Course Introducti... 40:08
02. The Crisis of the ... 48:37
View full playlist (22 videos)

Introduction to Psychology with
YaleCourses
1. Introduction 29:37
2. Foundations: This ... 53:19
View full playlist (20 videos)

YaleCourses: Philosophy of
mrchiltonsenglish
1. Course introduction 46:04
2. The nature of pers... 41:38
View full playlist (26 videos)

Figure 3.4: Searching for Yale University's courses in YouTube

R03 The videos are not versioned, and if a newer version of a video is uploaded, it replaces the old one. These modifications are also not indicated in the video page.

R04 The home page shows most recent and trending videos. Recommendations and popular genres are also shown.

R05 A search feature is available to search videos. The results are sorted based on relevance and popularity, while other sorting categories are also available.

R06 A like/dislike ratio is shown along with total number of views for each video. The view number is shown in the search result and homepage, but not the like/dislike ratio.

R07 Users can post their comments on each video. A like/dislike rating is also available to rate the video. Each comments can be further commented and given a like rating.

R08 Google collects information to improve its services. These can be adjusted in Google's privacy setting in their dashboard.

3.3 Facebook

Facebook is an example of a social networking site, with great e-learning potential despite not designed to do so. When Mark Zuckerberg launched Facebook in 2004, it started as a small social networking circle within Harvard. Now it is a multi million user social networking site with users all over the globe.

A study measures Facebook's potential as e-learning platform [37]. It noted Facebook's strength for collaborative work as follows:

- Simplicity and speed of creating and administering a work group
- Simplicity of use of native tools
- A high degree of external connectivity
- Internal expansion capacity
- Microblogging and lifestreaming features
- Mobile platform support

The study also points out Facebook's shortcomings for e-learning:

- Presence of 'noise' and distracting elements
- The drop-down comments system on walls tends to make it hard to see information
- Lacks a true system for tagging, filtering, searching and organising information
- Group discussion boards are too basic
- Lacks functions that are native to environments specifically oriented towards work groups
- No native synchronous bidirectional audio and video

Facebook's social networking tools can be used as e-learning tools. The fan page can be used as a course page, keeping all course related materials in the fan page. A multimedia post can be used to present course materials.

Facebook offers plenty of native socializing tools useful for gathering feedback. Comments can be posted to give qualitative feedback of the material. Those comments can be further liked or disliked to show general agreement with the comments. Posts can be liked or disliked to indicate its quality. Facebook also offers a poll tool to quickly gather majority votes. While lacking on native survey tool, many third-party developers offers Facebook integration to quickly use their service with Facebook credentials.

Facebook also gathers data from its user, and these information is not made available to the user. According to Facebook's privacy policy [38], Facebook collects:

- Content and other information provided when using Facebook, including: sign up for an account, create or share and message or communicate with others.
- Information about the people and group connected to the user and how the user interacted with them
- Information about purchases and transaction made when using Facebook's services
- Device information from and about the computers, phones or other devices where the user install or access Facebook's services
- Information when the user visit or use third-party websites or apps that uses Facebook's services

In summary, here are the requirements of the exchange platform (Section 2.5) and how Facebook implement them.

R01 Medias can be uploaded and shared via posts.

R02 A fan page can be created where access is restricted to their roles [39]. Each five roles, Admin, Editor, Moderator, Advertiser, and Analyst has different access to the page. For example, a moderator can send a message as the Page, while an analyst cannot.

R03 There is no content versioning.

R04 There is no specific list for shared contents. Older posts can be looked up by scrolling down the Facebook timeline.

R05 There is no feature for searching for a specific post or content in a fan or personal page.

R06 The number of likes are shown in each posts indicating how many person liked that post.

R07 Feedback can be gathered using Facebook's comment and like feature. Comments on post can be commented and liked too.

R08 Usage data when using Facebook is gathered, such as user interaction and device information. Analyzed data is not made available to the public.

3.4 Moodle

Moodle, Modular Object-Oriented Dynamic Learning Environment, is a course management system for online learning. Moodle has become a term of its own synonymous with a software package designed to help educators create quality online instruction. Moodle is based on socio-constructivist pedagogy design, meaning, its goal is to provide a set of tools that support an inquiry- and discovery-based approach to online learning [21].

Moodle is an open-source project, so its source code can be modified according to different needs. A Moodle server can host many courses, and can invite students from different universities to join its courses.

Moodle delivers its course material primarily by the lesson tool. The lesson activity module enables a teacher to deliver content and/or practice activities in interesting and flexible ways. A teacher can use the lesson to create a linear set of content pages or instructional activities that offer a variety of paths or options for the learner. Teachers can include a variety of questions, such as multiple choice, matching and short answer. Students can be directed to different pages depending on the lesson develops. A lesson may also be graded, with the grade recorded in the gradebook.

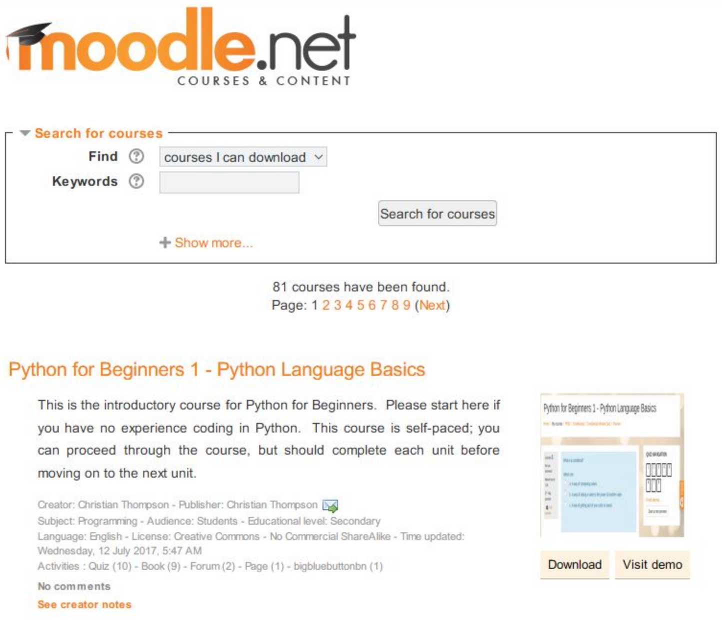
Moodle also has a feature that saves list of used questions in a quiz or lesson. The questions are saved in a question database so they can be reused. Questions, quizzes, and even whole courses can be exported. By using the backup feature, course related information can be saved, including comments, forum posts, lesson progress, and grades.

Moodle's question bank are not available for students. Moodle also has an extra steps to share questions to another course. Here is an example from Moodle's documentation about sharing questions to another course [40].

A Moodle site has three courses, Math, Physics, and Biology, all under Miscellaneous category. Fred, a teacher of Maths and Biology, creates some questions in his maths course and adds them to the quiz. This works since by default, questions are created in the question bank in a category of that course. Fred is a teacher in the course so he can create and add questions to the quiz. Now, Fred wants to reuse his math questions to his biology course. However, the biology quiz is not in the maths course, so the questions are not available. The Moodle admin then moves the questions into Miscellaneous category. Now Fred can't see his questions since he has no permission outside his courses. The Moodle admin creates a new role with permission to access the question bank, then assign it to Fred. Now Fred can see his questions and use them for his courses.

The disadvantage of this approach is the question sharing requires some setup beforehand, and each questions needs to be put to the right category. This could lead to some confusion. However, an extra layer of user can be an advantage if one is looking for a more fine-grained user access control. For example, a teacher with question sharing access and another without.

The Moodle community also has a website that allows Moodle users to share their courses and course contents, Moodle.net. Moodle.net serves as a place to find free to join Moodle courses from universities around the world.



81 courses have been found.
Page: 1 2 3 4 5 6 7 8 9 (Next)

Python for Beginners 1 - Python Language Basics

This is the introductory course for Python for Beginners. Please start here if you have no experience coding in Python. This course is self-paced; you can proceed through the course, but should complete each unit before moving on to the next unit.

Creator: Christian Thompson - Publisher: Christian Thompson
Subject: Programming - Audience: Students - Educational level: Secondary
Language: English - License: Creative Commons - No Commercial ShareAlike - Time updated: Wednesday, 12 July 2017, 5:47 AM
Activities : Quiz (10) - Book (9) - Forum (2) - Page (1) - bigbluebuttonbn (1)
No comments
See creator notes

Figure 3.5: Moodle.net search

Moodle provides many tools to gather student feedback. The choice activity module lets teachers ask a single question and offer a selection of possible responses. It can be used to facilitate student decision-making, for example allowing students to vote on a direction for the course. The feedback activity module lets teachers create a custom survey to collect feedback from students using a variety of question types including multiple choice, yes/no or text input. It can be used for course evaluations, helping improve the content for later participants.

Moodle comes with standard roles that can be assigned to its users. These roles have different access and permissions to the system. Those roles are [41]:

- Site administrator: can "do everything" on the site
- Manager: a lesser administrator role
- Course creator: can create courses

- Teacher: can manage and add content to courses
- Non-editing teacher: can grade in courses but not edit them
- Student: can access and participate in courses
- Guest: can view courses but not participate
- Authenticated user: the role all logged in users has
- Authenticated user on the front page role: a logged in user role for the front page only

Moodle is also extendable. Many Moodle plugins are available in Moodle's plugin directory that expands on Moodle functionality, for example, rating a course with a five star rating, or creating custom surveys, or a more detailed student statistics.

In summary, here are the requirements of the exchange platform (Section 2.5) and how Moodle implement them.

R01 Courses, exercises or quizzes are exported into an xml or archived format, which can then be uploaded to another course.

R02 Moodle allows specific roles to be able to change other specific role capabilities based on the context. For example, a teacher in a course may want all students (users with a student role) to be able to edit all forums in that course [42]. Custom roles can be added via admin interface. These roles can have different permissions and different access of the system.

R03 Courses and its contents are not versioned in Moodle.

R04 Moodle dashboard shows lists of available courses. Each course shows their materials grouped by either topic or week.

R05 Moodle provides a global search [43] to search everywhere on the Moodle site that you have access to. A student can search their courses for particular lecture notes, for example, or a teacher could search for subject-related activities.

R06 The result of surveys are shown only to the teachers and not to the student.

R07 Feedback can be gathered via choice and feedback activity. Further feedback methods can be implemented by installing plugins.

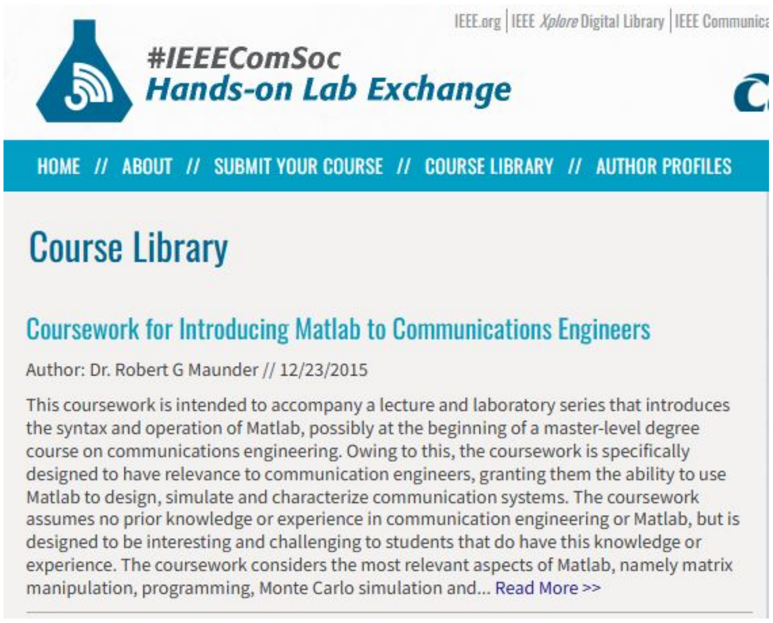
R08 Moodle analytics generates report for website use such as user logins, preferred language and comments. Engagement Analytics block provides information about student progress against a range of indicators and student activities which has been identified by current research to have an impact on student success in an online course [44]. Using Overview Statistics [45] plugin a report can be generated that produces various site and course report charts.

3.5 IEEE ComSoc Hands-on Lab Exchange

The ComSoc Hands-On Lab Exchange is designed to provide a mechanism for sharing course materials, projects, and best practices relevant to lab-based communications education [46]. The assignments shared in the ComSoc are published in two stages. First, they are available to the user base so that they can be peer reviewed. Second, once they are peer reviewed, they are marked as peer reviewed.

The ComSoc divides materials into two categories, lab experiments and lab courses. Lab experiments are similar to iLab lab modules, and lab courses are a collection of lab experiments. It also has a minimum submission (lab experiment), and must prove that the experiment has been used in a course, and a description of how the materials are improved based on the experience.

The available courses are accessible in their website [47]. It provides a search filter to help search the available courses. One course is also displayed on their main landing page as featured course to promote the course. When viewing the course, a description of the course is given. Available materials are divided into open access files that everybody can access, and instructor only files.


IEEE.org | IEEE Xplore Digital Library | IEEE Communica
#IEEEComSoc
Hands-on Lab Exchange
HOME // ABOUT // SUBMIT YOUR COURSE // COURSE LIBRARY // AUTHOR PROFILES

Course Library

Coursework for Introducing Matlab to Communications Engineers

Author: Dr. Robert G Maunder // 12/23/2015

This coursework is intended to accompany a lecture and laboratory series that introduces the syntax and operation of Matlab, possibly at the beginning of a master-level degree course on communications engineering. Owing to this, the coursework is specifically designed to have relevance to communication engineers, granting them the ability to use Matlab to design, simulate and characterize communication systems. The coursework assumes no prior knowledge or experience in communication engineering or Matlab, but is designed to be interesting and challenging to students that do have this knowledge or experience. The coursework considers the most relevant aspects of Matlab, namely matrix manipulation, programming, Monte Carlo simulation and... Read More >>

Figure 3.6: IEEE ComSoc Lab Exchange Lab Library

Feedback or review in ComSoc is twofold. The first is a checklist review to ensure that all of the requested materials are supplied. The second is an anonymous peer review. After the materials have fulfilled the two process, it is approved for publication and marked with an "IEEE ComSoc peer-reviewed" badge. An advantage to this process is that a submission is always checked for correctness, and has all materials needed. The explicit peer review badge also shows that the material is trustworthy and of a good quality. The author of the material also receives feedback from the reviewer.

In summary, here are the requirements of the exchange platform (Section 2.5) and how IEEE ComSoc implement them.

R01 Course or labs are submitted via a web form in ComSoc's website. The materials for the labs are then uploaded.

R02 The roles are divided into instructors and students. Students cannot access instructor only files and cannot upload labs.

R03 Materials are not versioned.

R04 The list of available labs can be accessed in ComSoc's website [47].

R05 A specific lab exercise can be searched from a search feature.

R06 After being peer-reviewed, a peer-reviewed badge is displayed on the course information.

R07 Feedback for lab exercises are gained from peer review.

R08 We could not find any information relating to learning analytics in their website, so we assumed they do not implement this feature.

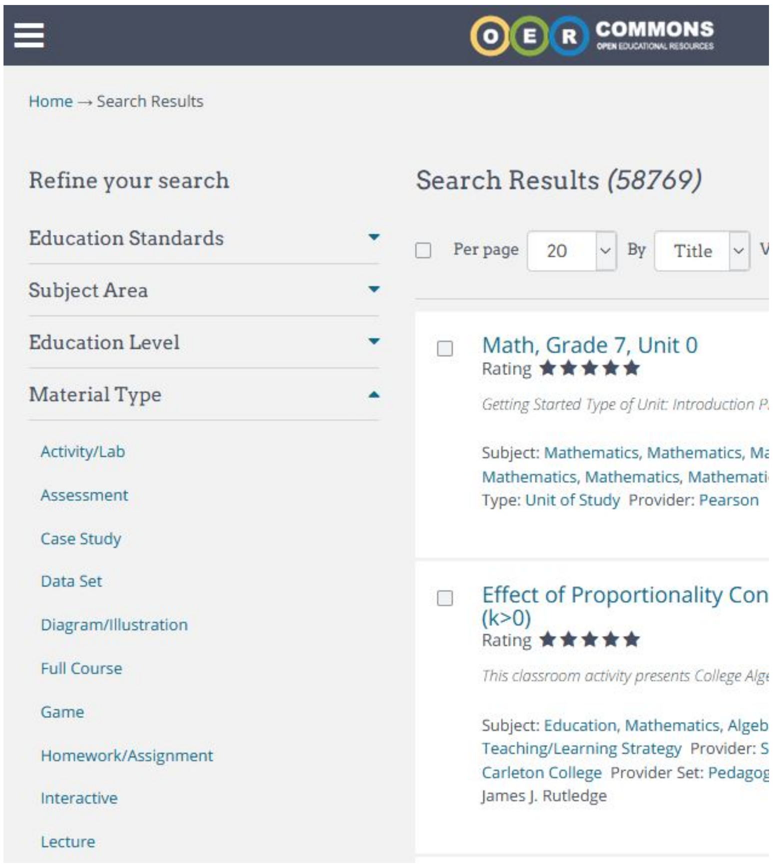
3.6 OER Commons

Open Educational Resources (OER) are teaching and learning materials that may be freely used and reused at no cost, and without needing to ask permission. Unlike copyrighted resources, OER has been authored or created by an individual or organization that chooses to retain few ownership rights. Resources can be downloaded and shared, or modified before reposting them as remixed work [48].

OER Commons offers many grouping categories to group its contents, such as subject areas, grade levels, and material types. Unlike IEEE ComSoc, OER Commons allows uploading the media files of the labs or courses without sharing the said labs or courses. The advantage of this approach is that materials can be shared in smaller units.

The materials can be accessed in OER Commons website [49]. The website provides keyword search, tags, group filters, and collections to help users find the materials they need.

OER Commons gathers feedback via comments and five star rating. An account is needed to post these feedbacks. Averaged rating of the material can be seen in the material list and their content page. The material specific page also shows comments on that material.



Search Results *(58769)*

Math, Grade 7, Unit 0
Rating ★★★★★
Getting Started Type of Unit: Introduction P
Subject: Mathematics, Mathematics, Ma
Mathematics, Mathematics, Mathemati
Type: Unit of Study Provider: Pearson

Effect of Proportionality Con
(k>0)
Rating ★★★★★
This classroom activity presents College Alge
Subject: Education, Mathematics, Algeb
Teaching/Learning Strategy Provider: S
Carleton College Provider Set: Pedagog
James J. Rutledge

Figure 3.7: OER Commons search feature

In summary, here are the requirements of the exchange platform (2.5) and how OER Commons implement them.

R01 Materials can be uploaded after logging in with an account. Different material types can be uploaded, such as labs, assessments, quizzes, lecture notes, or textbooks.

R02 Materials can be freely downloaded without a credential. An account is required to upload materials.

R03 Materials are not versioned.

R04 Available materials can be accessed from their website [49]. Grouping by multiple categories are available.

R05 Keyword search, tags, media types, and subject filters are available.

R06 Averaged star rating shown in entries in material list. Comments shown in material page.

R07 Feedback is gathered via comments and five star rating. An account is required to either comment or give a rating.

R08 Not mentioned in the website.

3.7 Summary

So far, multiple platforms are presented to discuss how they can provide solutions for the exchange platform requirements (Section 2.5). Here is a brief summary of the comparison and some interesting remarks about their implementation.

In terms of course materials, MOOCs provides the most structured materials by arranging them into courses, then topics, then materials for the topics. MOOCs also present their materials by combining them in one page, unlike YouTube that can only show videos. MOOCs also give a more guided approach to delivering the content, compared to ComSoc that only provides instructions and materials.

In terms of course catalog: these platforms each has a different implementation. Coursera offers a specialization (a group of related courses) to quickly find similar topic. MOOCs also offer group by academic fields. Some do not provide grouping, like Moodle.net that only show list of available materials. In all of them however, a search function is always provided. In social media like Facebook, it is harder to search for courses or any materials, since it is not originally meant as an e-learning platform.

Feedback is gathered using different means across these platforms. Quantitative feedbacks are gathered via ratings, while qualitative feedbacks are gathered via reviews, comments, or survey questions. Social media provide the necessary tools to gather these feedback, but not to process and aggregate them for quantitative analysis. In these platforms, feedback is gathered for the course (for e-learning platforms) and its materials. MOOCs always give a pre-course survey to gather some info of its participants, and another after finishing the course to gather feedback for the course experience. For a more specific feedback, for example for a lab exercise, ComSoc uses peer review, while other platform only gather feedback for entire courses. ComSoc is also the only platform mentioned here that does not use user review, but rather a peer review.

So how does the exchange platform compare with these platforms? Creating and uploading courses, lab exercises and questions, are the main features of this platform. Compared to MOOCs for example, this creates a community driven platform, where course materials are available for use.

Another defining feature of the exchange platform is versioning. By using versioning, different versions of materials can be tracked and offered. Lab exercises or questions can be forked to quickly create a new one using the original as template.

A feature lacking in the exchange platform is communication platform. Other platforms such as YouTube provides a better social media integration and communication with fellow students.

As a brief summary, here is a comparison table of different platforms with features relevant to this thesis. In all the listed platforms, no versioning is available.

Platform	Uploading materials	Material list	Material search	Feedback gathering	Usage data gathering	RBAC	Displaying feedback
MOOC	text, images, videos	course catalog	course search	survey, review, rating	site usage	teachers and students	latest review and rating
Youtube	videos	catalog and recommendations	keyword search, recommendations	comments and like / dislike	site usage	comment moderators	like / dislike ratio
Facebook	posts, events, comments, fan page	fan page	not available	comments, like / dislike, survey	site usage	Page roles with different permissions	like count, comments
Moodle	xml or zip	course and materials	global search	survey, comments	site usage and activity	custom roles with different permissions	not shown
IEEE ComSoc	lab exercises	list of labs	lab search	peer-review	not mentioned	instructors and students	peer-reviewed badge
OER Commons	multiple medias	multiple medias	keyword search, groupings, tags	comments, ratings	not mentioned	credentials required	comments, ratings

Table 3.1: Comparison table of e-learning platforms

Chapter 4

Design

Based on what we discussed so far in Chapter 2 and 3, possible components to construct an exchange platform are defined. Here the exchange platform is described in more detail.

As mentioned in the previous section, the exchange platform is based on the idea of sharing lab exercises. In the exchange platform, this idea is expanded to include course materials and quizzes.

The idea of the exchange platform platform, the *iLab*, only includes iLab courses (*iLab* and *iLab 2*). The exchange platform also includes other courses, not only *iLab* courses. These courses can then build upon uploaded materials from other courses in the exchange platform.

This concept can also be expanded to multiple universities or institutions. Each universities hosts their courses in the exchange platform, and share their course contents. By using a versioning system, a master version is kept, and a branch made for each universities. This way, each universities can have different version from the master template, and can modify it based on their requirements.

In order to provide a course sharing platform, the exchange platform needs a course creating tool to let the users create their courses, labs, and materials. Then these created materials needs to be uploaded into a common pool. Furthermore, these materials needs to be stored in a digital format. A format needs to be specified, and how they can be shared across the platform.

Since there are multiple courses, labs, and materials in the *labsystem*, the users need a way to find the one they are looking for. This can be done by having a catalog of available contents. A search or filter functionality can be made to look for a specific content.

One issue when hosting user created materials are quality control. The uploaded materials needs to be rated or reviewed to be appropriate or up to standards. A curation or review system is required to ensure quality of the uploaded materials. The exchange platform needs a feedback system from its user to rate its contents. Therefore in the exchange platform, students can rate and review the materials they did. This is discussed further in Section 4.1.6.

Course materials can change over time, from student feedback, updated information, or corrections. Therefore the system also keep tracks of changes of these materials. This can be done by using a versioning system. This is discussed further in Section 4.1.3.

Another important point to consider is integration with the current *labsystem*. The *labsystem* already provides some features (e.g. creating lab exercises) and is used by the Chair of Network Architectures and Services for the *iLab* courses, this is discussed further in Section 4.2.

In this chapter, more details of these requirements is discussed. Based on those discussion, possible implementation of the exchange platform is described, and how to integrate it with the *labsystem*.

4.1 Requirements

In Chapter 2 the following requirements for the exchange platform are defined:

R01 Upload materials for sharing (Section 4.1.1)

R02 Role based access control for sharing materials (Section 4.1.2):

R03 Materials versioning (Section 4.1.3)

R04 List available materials (Section 4.1.4)

R05 Search shared materials (Section 4.1.5)

R06 Display processed feedback (Section 4.1.7)

R07 Gather and analyze feedback about the materials (Section 4.1.6)

R08 Analyze and visualize usage data of the labs (Section 4.1.8)

In this section the requirements are discussed in more detail. Implementations from related platforms is discussed and compared with. Possible solutions can then be identified, compared, and chosen based on our requirements.

4.1.1 Uploading and Sharing Materials

In this section, we describe the requirements related to content management in more detail, look at other implementation from other platforms, and choose which to implement.

In the exchange platform, a lab is defined as a set of exercises in a course. A lab can have course materials given in the *labsystem* via text, images, or videos. A lab works as an assessment or exercise module. A lab can have multiple questions asked in multiple forms, for example, multiple choice questions, text, or number. A lab module or exercise can be imported into multiple courses. The lab module can be uploaded into the community website, making it available to the community.

Some platform packaged the shared material in a format, while other does not. Moodle export courses to an xml or archived file. ComSoc does not export the lab exercise into a format, but instead only offers its instructions and required materials, accessed from the lab website.

Exporting or downloading materials can be done similar to Moodle course export, by exporting them to an xml or archive file, which the current *labsystem* already has implemented. Another way is to let users download individual files of a lab, without downloading or exporting the whole lab. This also lets the versioning system keeps track of each individual files instead of the whole lab.

An upload feature is required to share the material to the platform. Moodle can import the exported courses or exercises to quickly use those in the Moodle server. Creating a new lab or course can be done via web forms: post a text instructions and attach the necessary files (like Facebook), or by also uploading the instructions (like ComSoc).

As previously described, the exchange platform is made to support multiple courses and universities. Each course can be made based on existing courses. Users should be able to choose a course and create a new course based on it. Different universities can also host different versions of courses.

Based on these points, and for other reasons from other requirements described below, we decide to use Git to host the files for the exchange platform. Lab materials can be uploaded to and downloaded from the Git repository as projects. Uploading contents can be implemented as commits or creating new project, and downloading as a simple download. Some Git implementation also provides downloading repository or folder as a zip file. Different versions of courses for universities can be done by Git branches. Labs can be downloaded as a template to create a new lab by forking the repository. Some Git implementations also provides APIs to construct a frontend website that access Git repository, separating presentation of the exchange platform from the Git repository.

Using Git repository to host lab files has disadvantages, such as limited file size [50].

Some Git providers also provides Git Large File Systems (Git LFS) to store large files by using reference pointers within small text files to point to large files stored on the GitLab servers [51]. However, this should not be a problem considering the expected file size of the labs.

4.1.2 Role Based Access Control

In this section, we discuss requirements for Role Based Access Control, and how they can be implemented.

The purpose of having a role based access control in the exchange platform is to define permissions of roles and assign users to specified roles. An RBAC is needed to prevent user from misusing the system by using features he or she is not meant to access, such as distributing solutions. Another advantage of having RBAC is to have a preset roles that have clearly defined permissions (for example, Youtube's moderator moderates comment section, but not video content) that are easy to understand and manage.

Looking at other platforms discussed so far, they have a preset of roles that have a specific permissions. For example, YouTube moderator only has access to the comment section. Moodle extends this further by allowing the creation of custom roles.

From Section 4.1.1, the exchange platform also hosts multiple courses from different universities. Therefore, the role access for each labs should be definable for each university. The labs feedback access should also be limited by participants of those courses, so each students should have different access or roles according to the courses. The site should also be accessible without logging in, with restricted access otherwise. Contents such as solutions should not be accessible without having sufficient permissions.

First we need to define the functions or features the roles can have access to:

- Administration: creating new user, assigning roles to user, creating new universities, courses and labs. For site administration.
- Uploading content: adding or modifying materials of labs or courses. For adding content to the platform.
- Downloading content: accessing materials of labs or courses.
- Feedback: for giving feedback to labs.

With these points in mind, we decided to use Git for RBAC of the exchange platform. Git can assign permissions to users in a project or repository. An advantage to this approach is projects containing restricted contents (such as solutions) can be restricted to certain users. Access to these contents should require authentication. Another advantage to using Git is that every project's access can be specified: it can be made public, or private, or to assigned members. This allows a more fine tuned access to labs hosted in the repository.

4.1.3 Version Control

As discussed previously, a version control system is needed to keep track of the changes in the exchange platform. In this section, we discuss the functionality needed from the versioning system and how it can be implemented.

The most basic functionality needed from the versioning system is keeping track of changes of contents in the exchange platform. For example, updates on course materials in a course for different semesters. The system should note the files being changed and its changes.

From Section 4.1.2, it should also keep track of changes between courses of different universities. For example, a university creates a new course (a same course) based on materials from another university. The system should note the differences between the two courses.

Another functionality needed is accessing previous versions of a course. For example, accessing an old version from a lab last year. The system should list available version from a version history and provides download or branching tool from that version.

Based on these descriptions, we decide to version the exchange platform with Git. The Git repository stores the questions, and keep track of the changes to the lab questions. Git branches can be used to separate the master lab files with questions from other universities or courses.

An advantage of using Git to version the labs is that changes to the lab contents can be tracked by Git. Each file in the Git project can be tracked separately, and each version from different commits can be accessed anytime. A disadvantage of using Git is since it stores every version of every file, Git requires every repository to have as much free space on a hard drive as consumed space at all times [50].

4.1.4 Listing Materials

In this section we describe the requirements for listing materials, how other platform implements these requirements, and choose which to implement.

The lab catalog shows courses available in the *labsystem*, and lab questions available in the lab pool. The catalog shows the ratings of each lab module, along with a brief description for potential student. Shown information should be helpful for potential students to help decide taking up the lab, such as, total time to complete, review, pre-requisite lab, etc. Teachers can also use the course catalog to search for interesting labs when they want to make a similar course.The labs should be sorted by their rating, with the best rated first, then by their date, most recent first.

MOOCs like Coursera hosts many courses from universities, and display these courses grouped by universities. The exchange platform hosts courses from multiple universities, but displaying these grouping is not necessary, since we would like to promote the courses regardless of their university affiliation.

Some MOOCs also provides grouping by specialization and areas. Since this enables users to quickly find courses of similar topic, this should be implemented in the exchange platform.

Open Educational Resources Commons (OER Commons) provides a list of publicly available resources. In addition to courses and labs, individual materials are also listed, such as lecture notes, charts, or textbook. This allows users to find materials from courses or labs or areas without downloading the whole courses or labs. While this is a great concept, this is not implemented in the exchange platform, since we focus on courses and labs, instead of their individual contents.

Based on previous discussions and analysis of other platforms, here are the points we choose to implement.

- List of lab modules. Shows all lab modules available.
- List of solutions. Only accessible if authenticated and have access to the solutions.

A search and sort feature are provided to help users search through the catalog. Similar to MOOCs, a brief description of each labs is shown in the catalog to provide a quick overview.

4.1.5 Searching Content

In this section, we discuss the requirements for searching content, how other platforms implements these requirements, and which one to implement.

To help users find content they want, a search feature is often given. This can be a simple search by keyword, grouping content, tags, etc. The exchange platform should provide search features to help finding a specific content and a more broad search to find related content.

A specific search is typically done with keyword search. A keyword search is done by typing a keyword into a textfield. An autocomplete is sometimes available to provide suggestions. An example of this is YouTube's search feature. A keyword search can immediately give the result the user wanted given the user knows what he or she is looking for.

A more broad searching or browsing feature is done in different ways, such as grouping, recommendations, tags, etc. Many platforms group their list of content, such as MOOCs (by universities or areas) and YouTube (by genre). Combined with recommendations, this can help users find related content. However, recommendations are outside the scope of this thesis, and is not implemented.

Based on these requirements, the methods we choose for the exchange platform's search function are simple keyword search and tag filter.

4.1.6 Gathering and Analyzing Feedback

From what we have discussed in the previous chapters, there are many different ways of implementing a feedback gathering system. Here are the feedback methods we choose **not** to implement and why:

- Peer-review. We choose not to implement peer-review procedure like ComSoc since we want to focus more on gathering student feedback.
- Interviews. Interviewing each student requires a lot of time and a survey can be used instead.
- Email survey. We want to get the feedback as quickly as possible and implement survey in the *labsystem* instead of using emails.

Here are the feedback methods we choose to implement and the reasoning behind it:

- Survey (Section 4.1.6.1). Survey or questionnaires is brief, and can get both qualitative and quantitative data before and after taking the course, and after finishing a lab module.
- Rating (Section 4.1.6.2). Ratings can be given to a shared material to give a quick score at a glance.
- Review (Section 4.1.6.3). Review is also implemented alongside ratings to give comments or opinions about the material to other students.

4.1.6.1 Survey

The exchange platform conducts survey for the same reason MOOCs does, to gather feedback about course content. MOOCs gather course survey multiple times, after enrolling and after finishing the course. The former is to gather data about the enrolled audience, and the latter is to gather course feedback.The exchange platform could do a similar method, but since we only want to gather lab feedback, only lab feedback survey is conducted. The survey should be done as soon as the student finish the academic activity [52].

Here are possible questions to ask for survey after finishing a lab course. The questions should be brief, and the whole survey should be finished in less than five minutes. The question should rate the lab module and general satisfaction of the module.

> How would you rate the module/lab? *[1-5 star]*
>
> Would you recommend this lab to other students? *[yes/no]*
>
> Are the contents of this lab interesting to you? *[very boring - very interesting]*
>
> What do you think about the time to finish this lab? *[too little - too much]*
>
> What do you think about the difficulty of this lab? *[too easy - too hard]*
>
> How much time did you spend on this lab? *[n hours]*
>
> Do you have any specific things you would like to cover ?

4.1.6.2 Rating

A five star rating is used to rate contents in the exchange platform. Other rating system considered include: ten star rating, like/dislike (YouTube, Facebook) or recommended/not-recommended (Steam). A like or dislike rating lacks a moderate option, and does not express how much the student like or dislike the rated item. A ten star rating is not used since a five star scale is simpler and easier to understand.

The five star rating system is chosen for its simplicity and its ease to be analyzed and produce an aggregated or average score. It can also be used to express indifference to the course (with score three). However, a study finds that in online reviews, moderate ratings are considered less useful than extreme ratings [53], which should be considered.

4.1.6.3 Review

A written review lets the student give qualitative comments on the content. It can show a more personal experience from fellow students that are otherwise only expressed in numerical ratings.

The review can be implemented like YouTube's comment system: simple comment to course or lab content. However, a comment box might encourage a shorter comments without much points. This is why review is implemented alongside ratings, to let students rate the lab, and write a review to justify their scoring.

Later the reviews can be displayed on the course landing page to provide useful information for potential students.

The whole feedback process and all three elements can be given in one survey. The question *How would you rate the lab? [1-5 star]* is a question that asks student to give their rating about the lab. A more general question like *Give a short review of your lab experience today* can be used to prompt a written review from the student. The rest of the questions are for survey or questionnaire purposes.

Here is the feedback form we choose to implement. Considering that we would like to integrate this with the *labsystem* and the *ilab* courses, an additional question is added for the *pre-lab*.

- How difficult was the lab for you? (1) Easy - (5) Difficult
- How interesting was the lab for you? (1) Boring - (5) Interesting
- How long was the lab for you? (1) Short - (5) Long
- How much time did you spend on the pre-lab?
- How much time did you spend on the lab?
- How would you rate this lab? (1 - 5 star)
- Do you have any particular feedback? Give us your comments

The student access to the survey needs to be defined. Only one submission is permitted to each student, for each lab exercise they participate. This requires a verification process before submitting a survey. There are a couple of ways to implement this.

- Require authentication before proceeding. This method requires an account made for each student, and the authentication might turn some people off.
- Token system. A token is generated for each student in each lab. The token is given to the student so that they can give lab survey. The tokens can also be associated with a lab, making it only usable for that lab.

For its advantages, we decide to use the token system for submitting feedback. Since the token is also associated with a lab, this also makes sure that the token cannot be used for another lab the student has not done.

The next point is the analysis of these survey result. Since question 1, 2, 3, and 6 are similar, they can be analyzed with the same method. Question 7 is not analyzed since we would like to read the comments as is.

For all the questions, descriptive statistics is used to describe and summarize quantitative information [54]. The statistics displays information such as mean, median, frequency, and standard deviation.

Once the feedback is collected, it must be stored before being processed. Similarly, the tokens for the feedback also needs to be stored and associated with their respective lab exercise. In the previous section, we decided to use Git to store the lab contents. The feedback result can also be stored in Git: it can be stored in a file and versioned using Git. An advantage of this approach is that no additional storage scheme like a database is required. However, as the result piles, this file gets larger and requires more overhead on Git. For example, adding a new survey result on a thousand line file requires Git to scan the whole file, which creates a performance issue as the file gets larger. Git API also requires the whole file to be sent, instead of just the text modification, which takes a lot of bandwidth.

For this reason, a database is made to store the survey result and tokens. Performance-wise it is more efficient than using Git, for both adding and retrieving entries. It also has an advantage of modifying the query to get a specific survey result. This database is described further in Section 4.5.

4.1.7 Displaying Feedback

In this section we discuss requirements for displaying feedback in the exchange platform website, how other platform implements it, and choose which to implement.

The processed feedback is displayed for each courses and labs. The feedback gathered by the exchange platform is discussed in the Section 4.1.6.

A simple score or star rating can be displayed in the catalog. Open Educational Resources Commons displayed star rating of the item in their content catalog. On the other hand, some platform like Moodle and YouTube does not show the item ratings or score in the content catalog. We choose to show the ratings in the catalog to give users a quick summary of the item's quality.

A more detailed rating can be shown when viewing the item, for example in a video page in YouTube or material page in OER Commons. These individual pages should show the summary from processed feedbacks. From Section 4.1.6, we use surveys, ratings, and reviews to gather feedback. Some metric from learning analysis can also be shown.

The metric gathered from surveys are meant to help improve the content, and not relevant to interested users viewing the content. The average score from ratings should be shown prominently as the primary score of the item. Reviews should also be shown, with the latest shown first.

From the previous section, we described survey as the primary method of gathering feedback in the exchange platform. Since the survey questions ask different questions with different types of inputs, these questions are analyzed and visualized differently.

The first three questions of the survey asks opinion regarding the lab, presented in a likert scale. There are several methods of visualizing data from a likert scale, such as a simple table, bar chart, diverging stacked bar chart, grouped bar chart, or divided bar chart [55]. There are also many variants for diverging stacked bar charts [56]. A divided bar chart is also replaceable with a pie chart.

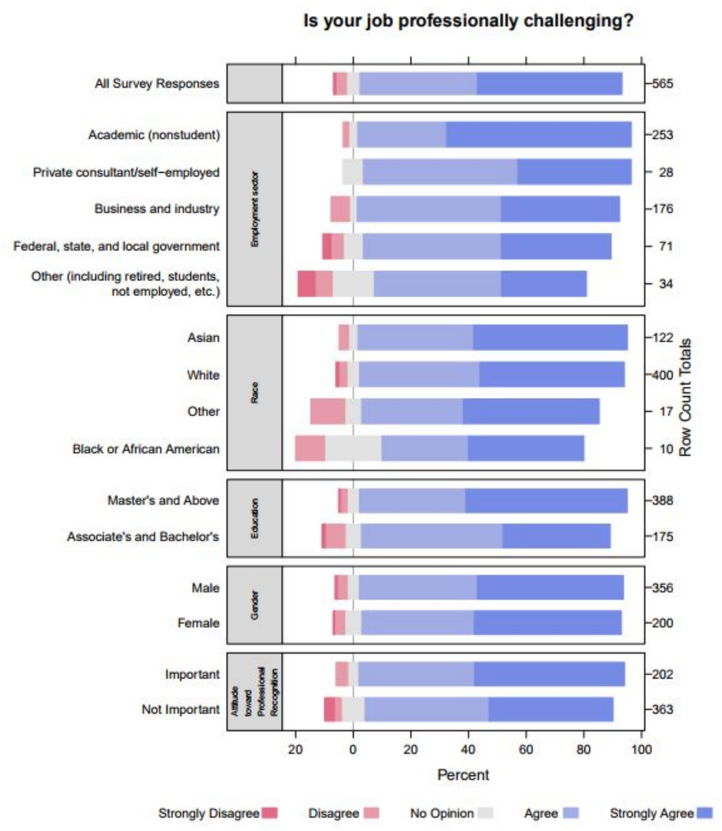


Figure 4.1: Diverging stacked bar chart

However, a diverging stacked bar charts is meant for questions with the same answer scale. In our survey, the questions, while being expressed in a likert scale, is presented with different scales. Question one is easy-difficult, question two is boring-interesting, and question three is short-long. Unlike in a diverging stacked bar charts where the all the questions have the same answer options. Therefore a diverging stacked bar chart is not be used.

Another alternative is to use a divided bar chart to visualize the proportion of each answer compared to all submissions. A pie chart can also be used in this case. However, divided bar charts can be grouped together into a single chart, which saves space and provides an overview to all questions. For this reason, a combined divided bar chart is used as the primary visualization of the first three questions.

The combined divided bar chart however, does not show detailed information like mean, median, or standard deviation in the chart. A bar chart can be used to show these detailed information as annotations in the chart. Since we want the main chart to

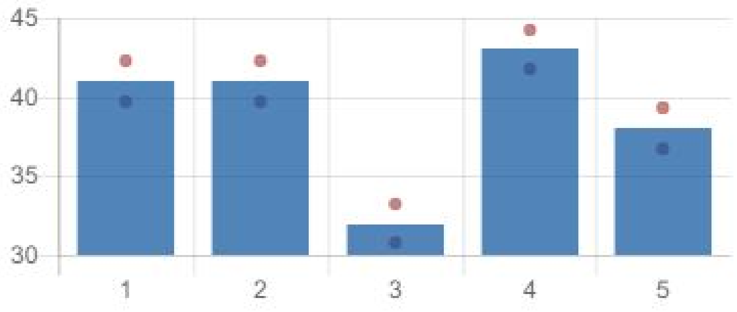


Figure 4.2: Annotated bar chart

give an overview of the answers, while also providing a more detailed visualization, a separate chart for each of these questions is also provided. This is made in a bar chart, and it visualize a breakdown of the answers of the respective question with means, median, and standard deviation annotated in the chart.

This visualization cannot be used on the next two questions however. These questions ask the amount of time spent on pre-lab and lab, with a number (of hours) as an answer. Here a histogram is used instead, where x-axis shows the number of hours submitted, and y-axis shows the frequency of that answer.

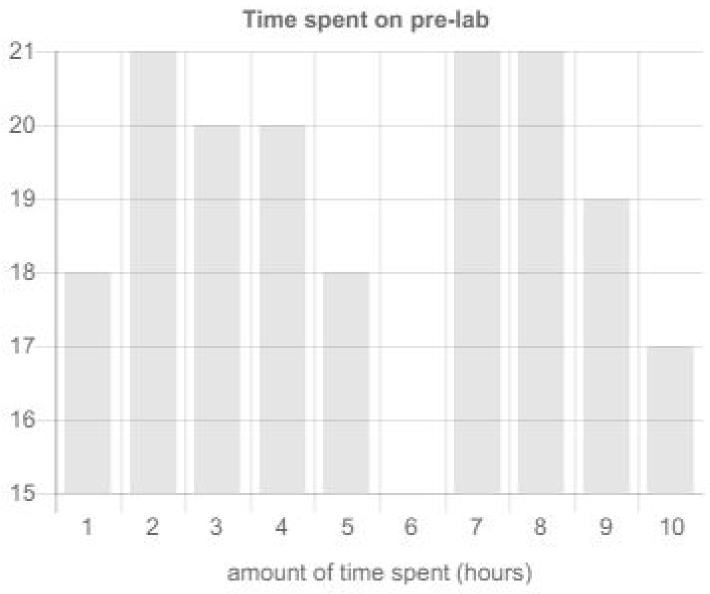


Figure 4.3: Histogram bar chart

Question number six asks a rating for the lab, given in scale one to five. Similar to the first three questions, this question can be visualized in the same manner. The visualization of this question is combined with the visualization of the first three chart.

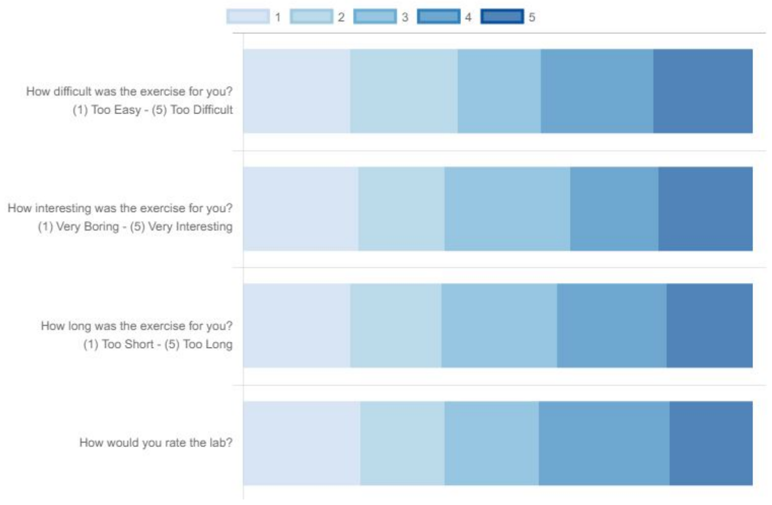


Figure 4.4: Horizontal stacked bar chart

Finally, the last question asks for a comment or review of the lab. One method of visualizing comment is tag cloud or word cloud. A word cloud displays a list of words, keywords, or tags in a cluster. A tag's popularity is expressed by its font size, and its other properties like color can also be manipulated to further make it stand out [57]. It can be used to identify trends and patterns in the comments.

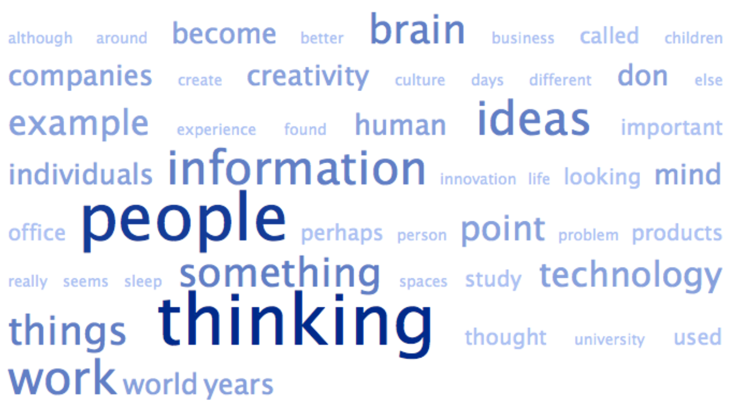


Figure 4.5: Tag cloud

However, tag cloud only emphasizes frequency of words, not necessarily its importance. The context of the word is also lost in the visualization. A research introduces a visualization method that preserves its context by using a two-level visualization with a trend chart [58]. Since the comments are not analyzed, they are displayed as they are without analysis. Further works in the portal may include this visualization.

4.1.8 Gathering, Analyzing, and Visualizing Usage Data

The exchange platform shows analytics of the labs. The usage and analytics is gathered by the *labsystem* as the students use the system to do lab exercises. The gathering method and metrics gathered by the *labsystem* is outside the scope of this thesis. It is assumed that analysis data of each labs are already provided. These are then analyzed and visualized in the portal.

At the time of writing, the *labsystem* gathers usage data when students perform lab exercises. An example of this is tracking grades of each questions to show the pitfall questions where most students lose credits. Another example is tracking lab completion time during exercises.

These analysis are gathered and performed in the *labsystem*. The results of these analysis can be stored and shown in the portal website. Here are example lab module metrics for analysis, their required inputs, possible applications, and example visualization.

- Completion time: amount of time to finish the lab module.
  Required input: start and end of the lab exercise. Lab completion can also be tracked across multiple login sessions, in that case, also requires login and logout dates.
  Purpose: used to show average amount of time required to finish the lab exercise. Can be helpful when a student wishes to choose a lab exercise based on time constraint.
  Visualization: a bar chart showing the list of lab exercises, and their completion time

- Question completion time: amount of time to complete each question in a lab exercise.
  Required input: start and end of the lab question. Page visit duration and amount of page views can also be used to estimate time on a question, assuming one question shown in a page.
  Purpose: used to show average amount of time required to answer a question in a lab. Can be used to show time consuming questions
  Visualization: a bar chart showing the list of lab questions in a lab, and their completion time

- Average grade: average grade of students taking the lab exercise.
  Required input: scores of every participant. Can be filtered by different criterias, such as different academic term, different lab version, or different faculty or university.
  Purpose: track problematic questions, i.e. where most students loses points, shows average score of participants. Later, this can be shown along the lab, in the lab list.

Visualization: a bar chart showing average grades of each lab questions

We have now described the requirements of the exchange platform. Before possible implementation is considered, the currently running *labsystem* is examined in the next section.

4.2 Labsystem Workflow and Integration

The *labsystem* is an e-learning software used to deliver the *iLab* courses. The system running *iLab* can be accessed in `https://ilab.net.in.tum.de`. The source code of the *labsystem* is available in `https://github.com/m-o-p/labsystem`

The *labsystem* offers many features, and we only discuss those needed for our requirements. Let us see how the *labsystem* compares to our requirements.

- Uploading materials: labs can be exported and imported using import/export tool.
- Role based access control: students can only perform lab exercises and give feedback.
- Versioning: there is no versioning.
- Content list: list of labs on the sidebar. Labs from previous semesters are also accessible. Also offers a calendar/schedule feature.
- Content search: not available
- Display feedback: not available or shown to students
- Gather feedback: after each lab, via text comment.
- Analytics: tracks multiple metrics on each lab, such as completion time, credits earned in each questions, average grade.

Since the lab system offers these features already, let us consider how to expand it to fulfill our requirements.

- Uploading materials: already sufficient.
- Role based access control: already sufficient.
- Versioning: labs are stored in a database. Since we decided to use Git for versioning, this requires either storing multiple entry in the database (and indicate their Git commit hash), or storing only the commit hashes in the database and access them from Git.
- Content list: already sufficient.
- Content search: a search feature needed.

- Display feedback: not available or shown to students
- Gather feedback: survey and ratings feature needed.
- Analytics: displaying gathered analytics feature needed.

Expanding the *labsystem* to offer our requirements is possible, but it has some problems. First, if each *labsystem* installation offers its own lab catalog, there won't be any centralized place to share and obtain labs from. For example, there are three *labsystem* installation in three universities, each with their own lab list. If one is looking for labs, one has to look in three different sites. Processing feedback from the same lab is also difficult since it is distributed across multiple sites.

Using the *labsystem* as the exchange portal is possible but poses many problems. Instead, a separate portal site should be considered. A great example here is Moodle. The Moodle ecosystem has similarities to our *labsystem* and requirements. Each university or institution can run their own Moodle instances. Each of these instances might offer the same course. This course can be shared to Moodle's content portal: Moodle.net. By sharing courses from Moodle.net, these multiple instances of Moodle can host the same course. This also centralizes the content list, with only one place to visit. Feedback can be given and processed in a centralized site.

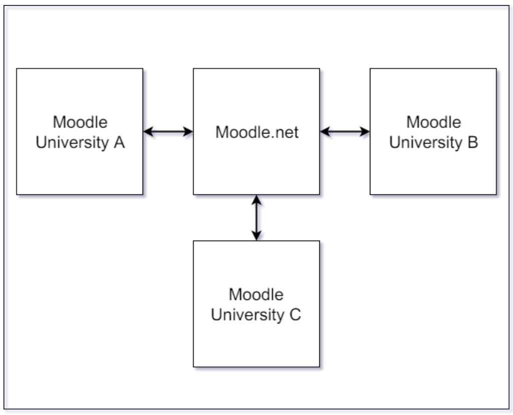


Figure 4.6: Sharing content between Moodle instances with Moodle.net

With this inspiration, a separate website can be made to serve as the primary content portal between *labsystem* instances. The portal shows available labs created by the *labsystem*. Lab feedback is also submitted to, processed, and displayed in the portal. It is for these reasons that we decide to make a separate exchange portal instead of expanding the *labsystem*.

Regardless, the *labsystem* still has features that we want and should not be disregarded completely. Instead, an integration with the portal should be considered. Since the *labsystem* has lab creation, import, and export tool, the labs should be created in, and exported to the portal from the *labsystem*. After a lab is exported to the portal, it is versioned there using Git (discussed further in Section 4.3). The lab catalog should be offered in the portal, since it hosts the labs from the *labsystem* instances. Feedback should be submitted to and processed in the portal to avoid processing from multiple instances. In the same manner, result of the feedback should be displayed in the portal. Analytics are gathered in the *labsystem* as the students perform lab exercises, and result of the analytics submitted to the portal where it can be displayed and visualized. More details on the integration can be found in the next chapter.

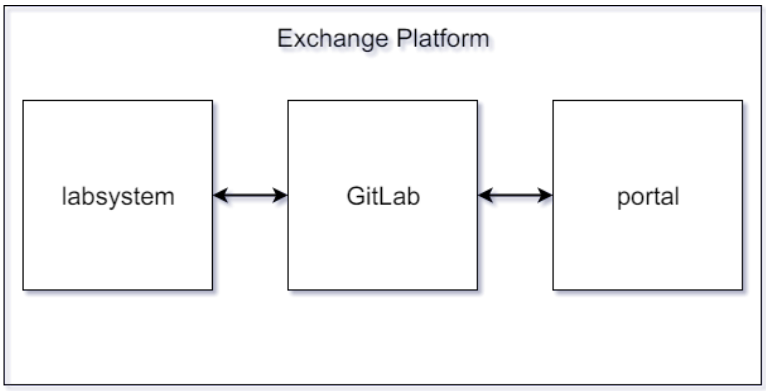


Figure 4.7: Sharing labs with exchange platform

Now that we have defined the workflow, and the role of the *labsystem* in it, we discuss the portal. As discussed in Section 4.1, Git is used to store the labs and version them. In order to show the contents stored in Git, a frontend website is needed. So the portal has two elements, the Git backend, and website frontend. We first discuss the Git backend.

4.3 Git

As mentioned in the requirements sections, the exchange platform uses Git to store lab contents and solutions. In this section, we discuss the features of Git that we use to construct the exchange platform, how to structure Git to store lab contents, define permissions, and how it communicates with the frontend.

4.3.1 Git Hosting

Before we can discuss about how to use features offered from Git, we must first decide which Git service we choose. There are many Git hosting services available. They provide code hosting and versioning using Git, and offers their own services on top of it to support development, such as issue tracking, merge request rules, or wiki pages. Here we compare two Git services, GitHub and GitLab.

GitHub is an online code hosting service based on Git, that provides an open development environment and visibility of projects through notification and a simple interface. This transparency promotes increased awareness and reduced communication [59]. In recent years, GitHub has become the largest code host in the world, with more than 5M developers collaborating across 10M repositories [60].

GitLab is a web-based Git repository manager with wiki and issue tracking features developed by GitLab Inc. According to a survey [61], GitLab has two-thirds market share in the self-hosted Git market [62]. It is one of the 30 highest velocity open source projects, according to Cloud Native Computing Foundation [63]. GitLab comes in Community Edition (CE) and Enterprise Edition (EE).

While the two are both web-based Git repositories, they offer different features. Here are the key differences between the two relevant for this thesis:

- Permissions. In GitLab, permissions are set according to people's role, rather than GitHub's read or write to repository. This allows GitLab to set a more flexible permissions, such as separating access from issue tracker with the code base.
- Self Hosting. GitLab CE and EE are available for self-hosting via many platform (Omnibus, Docker, Google Cloud Launcher). GitLab CE is available for self-hosting for free, while GitLab EE and GitHub Enterprise Edition is not [64].

Another point to consider is that the Git repository offered by LRZ for TUM is hosted in GitLab. For future implementation and integration with the LRZ GitLab repository, it is an obvious advantage to use GitLab at this point, to prevent or avoid compatibility issues that might rise.

GitLab and GitHub offers very similar features for source code hosting, but due to the ease of self-hosting and later compatibility with LRZ repository, we choose to use GitLab to serve as our Git repository.

GitLab is offered in two edition: Community (CE) and Enterprise (EE) Edition. GitLab CE is available for free, and GitLab EE cost scales based on the number of users. Projects hosted on Gitlab.com comes with GitLab Enterprise Edition.

GitLab CE offers:

- Unlimited amount of private or public repositories
- Built-in Continuous Integration / Continuous Delivery for free
- Issue tracker and comments
- Activity stream of the repository
- User roles based on permission [65]
- Protected branch. Set rules and permissions specific to a branch.

GitLab EE offers all the features of GitLab CE and more:

- LDAP and Kerberos integration. Allows authentication using LDAP or Kerberos.
- Merge approvals. Merge requests can be set to require a set number of approvals before accepted. Members of the project can be assigned as reviewers and merge approvals can be set to require approvals from these members.
- Contribution analytics. Offers an overview for activity of issues, merge requests and push events.

GitLab EE features are offered for enterprise needs, to support big organizations or projects. Considering the scale of the project, and the features offered, we choose to use GitLab Community Edition. If the need arise, the GitLab installation can be upgraded to Enterprise Edition, in addition to applying an Enterprise license.

4.3.2 Git as Backend

After deciding to use Gitlab, the next question is: how do we use Git to save the lab files for the backend? As mentioned earlier in this chapter, we construct the exchange platform with Git as a backend, and a portal website that accesses that backend.

GitLab provides a RESTful API [29] to access its Git repository without using the command line interface or its web interface. This allows the creation of applications that access the Git repository and its functionalities and features. The API lets the GitLab repository to act as a backend server for the exchange platform website. More is discussed in Section 4.4.

Now that we can access GitLab contents without using its web interface or command line interface, we can discuss how to arrange and store the lab contents. Git arranges its contents into projects. There are a few possibilities to store labs in Git.

The first is to place the labs into one project. This one project contains all the labs and solutions which can be arranged into hierarchical folders. For example we create a project `exchange_platform` with the following structure:

```
[project] exchange_platform
  – [folder] Labs
    – [folder] Lab_1
    – [folder] Lab_2
  – [folder] Solutions
    – [folder] Lab_1_solutions
    – [folder] Lab_2_solutions
```

Another alternative would be placing the solutions in the same project of the labs:

```
[project] exchange_platform
  - [folder] Labs
    - [folder] Lab_1 (with solutions)
    - [folder] Lab_2 (with solutions)
```

These configurations can be beneficial if there is no access restriction required. However, if one wants to restrict access to certain labs, it is not possible to do so here. In Git, once given access to the project, the whole directory can be accessed, i.e. restricting access to subfolders is not possible. This exposes the solutions to anybody with read access, which could give students access to the solutions when doing said exercises. Another disadvantage is version history in the project is combined from all the labs. Creating branches or forking the project also duplicates the entire project, which can be undesirable if one only wants to create a different version of one lab.

The second possibility is to arrange them to branches. A branch is created for each of the labs, and its solutions. The *master* branch contains a merged content from all branches. The structure would look like:

```
[branch:master] exchange_platform
  - [folder] Lab_1
  - [folder] Lab_2
  - [folder] Lab_1_solutions
  - [folder] Lab_2_solutions

[branch:lab_1] exchange_platform
  - [folder] Lab_1

[branch:lab_1_solutions] exchange_platform
  - [folder] Lab_1_solutions

[branch:lab_2] exchange_platform
  - [folder] Lab_2 (with solutions)
```

Using GitLab push rules, the branches can be protected so that only a specified role or users can push to that branch. This way, it can restrict write access to these branches. Read access however, is still unrestricted, since a user with a read access to the repository can still access the content of all the files in all branches. Versioning is better than the previous method, since GitLab can filter commit history by branch. However, this method also creates a lot of branches to manage.

The third possibility is to arrange each labs and solutions into their own projects. A project is created for each labs with/without the solutions. The structure would look like:

```
[project] Lab_1
[project] Lab_1_solutions
```

Since access is given to each project, the access of each user or each project can be specified individually with this schema. The advantage of using this schema is that access to each project can be specified individually. Creating a different version of a lab is also easier this way, by either creating a branch, or forking the project. However, if a lab depends on a content from another lab (meaning from a different project), one has to take care to maintain both project. For example, in the previous point two labs can be committed with one commit, while in this schema, two commits (to two different projects) are needed. The solutions can be placed either in the same project, or created as a separate project. When placed in the same project, it is easier to maintain, since changes to the lab and solutions only need one commit. However, this would mean that giving access to the project also gives access to the solutions, where if they are in a separate project, access can be granted separately. Therefore, we decided to use the third schema, where each labs and solutions are arranged into separate projects.

GitLab also offers a namespace system called Groups [66]. It group projects into directories and give users access to several projects at once. By default, when a project is created it is assigned a namespace from the GitLab user that creates it. When it is assigned to a group, it also assign a namespace to that project. For example, when project `portal` is assigned to group `frontend`, it is named `frontend/portal`. `GitLab 9.0` introduces Subgroups, allowing up to 20 levels of nested groups to help organize larger projects, and manage its users.

A user can be given access to a project by giving the user access to the project, or its group. When given access to a parent group, the user also has access to its subgroups. Groups can be used to organize types of lab contents, such as labs and solutions. The labs projects can be grouped into `labs` and solutions to `solutions`. Users can then be given access to the groups, and have permission to access its projects. This has an advantage of simplifying access restriction to the parent groups, e.g. student accounts can only access `labs` but not `solutions`. Access for the child projects or subgroups are inherited from the parent group.

We decided to use the groups to group labs and solutions project, in order to separate access control. Separate groups are made, a `labs` group containing the lab files, and `solutions` group containing the solutions of the labs.

Now that we discuss possible arrangements and access control of the lab projects, we discuss what access the users have in their assigned projects.

GitLab users have different abilities depending on the access level they have in a group or project [65]. If a user is both in a group's project and the project itself, the highest permission level is used. When giving project or group access to a user, its role in that project or group is also defined there. GitLab offers five non customizable roles:

- Guest. Can create issue in the issue board, leave comments, and see wiki pages.
- Reporter. In addition to having Guest's permissions: can pull source code, download code, manage issue tracker, and see commit status.
- Developer. In addition to having Reporter's permissions: can list and create merge requests, create and push to (non-protected) branches, create and maintain wiki pages.
- Master. In addition to having Developer's permissions: can invite new members, push to and enable protected branches, change project settings.
- Owner. In addition to having Master's permissions: can switch project's visibility, transfer project to another namespace, and remove project.

Accounts requiring access to the labs should be given at least Reporter permission, to access the files in the project. Developer access can be given to let the user modify the content of the project.

4.4 Frontend Portal

Now that we have discussed the possible backend structure with Git, we discuss the frontend portal. The portal is a website that acts as a portal to access the GitLab repository. This allows user to access content stored in the Git repository without using GitLab web-interface, and customize features based on our needs.

The portal shows the contents in the GitLab repository. Detailed description of the items is shown when needed. Feedback should be available for each lab modules. Users should be able to give feedback on each labs. Instructions to access the lab files should be made available in the portal. When available, solutions should be indicated and made accessible for authenticated accounts with sufficient permissions.

Based on these descriptions, the portal requires two minimum pages:

- Catalog: lists labs and solutions
- Description: description page for individual lab. Shows metadata and comments of that lab.

Before any information can be shown in the portal, it needs to communicate with the GitLab backend. There are a couple of methods to send data between the portal and the backend.

The content catalog can be implemented with a file containing a list of items (labs or solutions) and their information. This file should be updated after every content update or new content in the repository to keep it up to date. This file can be either stored in the web server that hosts the frontend portal, or saved in a Git project. In case of latter,

```
GET /projects/:id/repository/files/:file_path
```

can be used to get the file. An example file (in JSON) would look like:

```
[
  {
    "name": "iLab 1",
    "description": "iLab 1 labs",
    "awesome": true
  },
  {
    "name": "example lab",
    "description": "example lab",
    "awesome": false
  }
]
```

Another method is to use GitLab APIs. If the lab projects are arranged in projects-per-labs structure as mentioned in the last section, the endpoint

```
GET /projects
```

can be used to get the list of projects, hence the list of labs. In case all labs in one project structure is used,

```
GET /projects/:id/repository/tree
```

can be used instead to get the item list of the project. An example of `GET /projects` is as follows (not all attributes are shown):

```
[
  {
  "id": 1,
  "description": "my project",
  "default_branch": "master",
  "visibility": "private",
  "ssh_url_to_repo": "git@example.com:myuser/myproject.git",
  "http_url_to_repo": "http://example.com/myuser/myproject.git",
  "web_url": "http://example.com/myuser/myproject",
  "owner": {
    "id": 3,
    "name": "myuser",
    "created_at": "2013-09-30T13:46:02Z"
  },
  "name": "My Project",
  "name_with_namespace": "myuser / myproject",
  "path": "myproject"
  ...
  }
]
```

The advantage of the former is that it can include customized attributes. Since GitLab APIs return a fixed set of attributes, another method is needed for this. However, this file needs to be maintained and updated accordingly. Also, the path or the of the file needs to be known beforehand if GitLab API is to be used to fetch it. The advantage of the latter is that it requires no additional maintenance, since the GitLab APIs return list of projects (or files depending on the schema chosen). However, after authentication, the API also return the lists of all projects available for that user, including his private projects that is not relevant to the portal. The portal has to handle this accordingly.

A solution to this is to use Groups API instead of Projects API. Using

```
GET /groups
```

to get list of groups the user has access for, we can then use

```
GET /groups/:id/projects
```

to get the projects associated with that group. Now, this still has the previous problem of also returning list of groups unrelated to the portal, but this can be solved by creating a whitelist of groups parsed by the frontend. For example, the frontend can ignore every other group except *labs* or *solutions*.

With that in mind, the portal uses the method just described to retrieve the catalog. It doesn't require an additional catalog file to be maintained. The text file solution also doesn't handle authentication well, since it either uses the same file, or having to handle different file for different users.

The portal also need to show information of those labs. First lets look at *labsystem* lab file structure. The generated lab has the following structure:

```
[folder] css
[folder] data
  - [folder] withSolutions
  − readme.txt
[folder] files
[folder] images
preview.html
```

Similar to lab catalog, lab description for each lab can be stored and parsed from a text file. In this case, either the `readme.txt` from the data folder or `readme.md` from the project readme can be used. Alternatively, GitLab API provides a description of a given project via

```
GET /projects/:id
```

The advantage of using a file to store the lab file is similar to the catalog argument. The *labsystem* also already uses this schema. However, the path of the file containing the metadata needs to be known beforehand. Using GitLab API removes the need for maintaining this file, and in case all given attributes are sufficient (no custom attributes needed) no further implementations are needed.

The portal uses GitLab API to retrieve the metadata of the lab. The reason is that using a file adds to the required maintenance, and the path or id of the file needs to be specified to the portal, further adding complexity. Using `description` attribute of the GitLab `GET /project/:id` is sufficient for our need.

The portal can provide a compressed format of the project as a way to export the lab via

```
GET /projects/:id/repository/archive
```

Alternatively, the branch name can be specified to get that branch instead of the master branch. In case the one-project-contains-all-labs schema is used, this downloads all the labs. There is no API available to selectively download a specific folder (specific file is available), so this download is a disadvantage for that schema.

However, since we want to encourage using Git to access the lab, the frontend portal does not provide a download feature. Instead, a code instruction to clone the repository is provided.

GitLab comes with several feedback tools that we can use. Issues can be created for each project, and comments can be posted on those issues. A star rating is also provided after authentication to bookmark a project. The number of stars of a project is shown on its project page.

GitLab issues provides a platform for group members to communicate, share, and discuss proposals before and while implementing them [67]. Comments are available in GitLab, but it is implemented on issues, not projects. Users cannot comment directly on projects, but rather on issues on that project. Therefore an issue is be created for each project to serve as a comment hub. The frontend then shows the comments of the issue in the detailed page of that lab content. A simple feedback form can also be provided to submit comments. However, this method requires authentication.

The portal uses the Issue and Notes API to post comments to the lab projects. Using

```
GET /projects/:id/issues
```

the list of issues of that project is obtained, and then comments of that issue can be obtained via

```
GET /projects/:id/issues/:iid/notes
```

New comments can be posted using

```
POST /projects/:id/issues/:iid/notes
```

An advantage of using this method is that the comments are also visible in GitLab web interface, in case it is needed. However, `GET /projects/:id/issues` also returns other issues of that project. This can be handled by specifying a fixed name for each issues that hosted the comments. For example, for lab *iLab1*, a postfix *_comments* is used to create an issue named *iLab1_comments* to host its comments.

Considering that an GitLab issue might be used in order to track an issue or a problem for the lab project, creating this comments issue in the same lab project might add an unnecessary item to the issue list. For this reason, a separate project group is made to specifically host the comments. Each project uploaded to the Git will have a their own feedback project in this feedback group. This adds up to three projects created in the GitLab repository for each lab uploaded (labs, solutions, feedback).

There are a couple of issues in the system we describe so far. First, some features requires authentication. Posting comments on issues requires one, since the comments are posted as the authenticated account. Second, GitLab enforces a API limit for unauthenticated request. Requests without authentication are limited to 1000 requests an hour (GitHub also limits unauthenticated requests, and is lower compared to GitLab's request limit). And finally, GitLab's access controls are also better controlled for authenticated users.

These problems can be solved by creating a specific GitLab account for the frontend portal. The frontend GitLab account can be assigned to the projects that we would like to make publicly available in the portal. For example, the frontend can be assigned as member of lab projects, while making the projects' access members only. This way, the frontend can list the project in the portal, while still restricting access to the repository.

Another advantage of this method is that access exposed to the portal can be easily maintained by granting their access to the frontend portal. For example, we would like to show *Lab 1* in the portal, but not *Lab 2*. In this case we can set access to these projects as members only, then include frontend as a member of *Lab 1* but not *Lab 2*.

GitLab also has limitations. As mentioned in Section 4.1.6, storing survey result and token can create performance problems, since Git is not effective in handling large files. Instead of storing these in Git, a database is used instead. This database is described in the next section.

4.5 Database

In Section 4.3, we describe how to use GitLab to store the lab exercises. Due to Git's limitation in handling large files, survey results and tokens are not stored in Git, but in a separate database.

Since only the survey results and tokens are stored in the database, we keep the database as simple as possible. The lab information related to the survey is not stored in the database. One database is required, and two tables are made: one for survey results and one for survey tokens.

The survey result table only needs to store the answers. Since all the lab survey uses the same question, the survey question of the survey result does not need to be stored. The answers are saved in separate columns in the table. To associate the survey data with the lab exercise, the GitLab project id of the lab exercise is also stored. Since the lab exercises are versioned, the version of the lab is saved as well. The branch name and commit id of the lab exercise are used for this purpose. Additionally, the tokens are given timestamps to indicate their expiration date. This expiration date is also saved in the database.

The survey token table stores the token for submitting lab survey. The token is randomly generated, and assigned each to one student, for one lab exercise. Therefore the GitLab project id of the lab exercise is stored to associate the token with that lab exercise, so that the token can only be used for that lab. The version of the lab, however, does not need to be stored, since the token is valid regardless of the branch or version of the lab exercise. In order to clear unused token, each token is given expiration date. In the portal, the token is rejected once it passed its expiration date. This table should be periodically checked to clear expired tokens.

The database is the last component of the exchange platform. To summarize, the exchange platforms consists of the *labsystem*, a GitLab repository, a database, and a portal website. The *labsystem* imports and exports labs to the GitLab repository. The GitLab repository stores labs and solutions. The database stores survey token and data. The portal shows the list of labs and acts as a feedback hub for the labs.

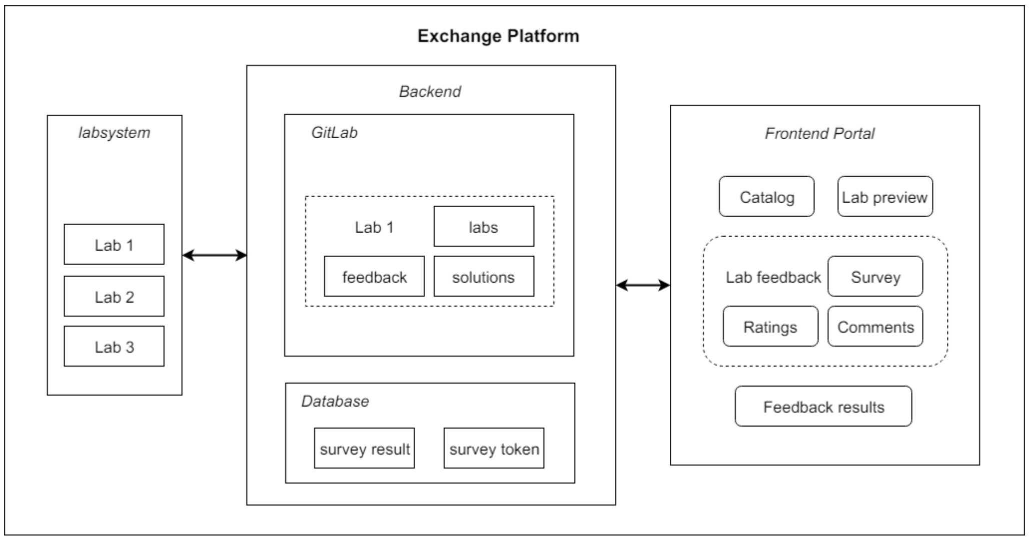


Figure 4.8: Exchange Platform workflow

So far, we have described all the requirements, how to create an exchange ecosystem based on labs created by the *labsystem*, how to use GitLab to store labs, and how to construct a website to serve as a content portal. In the next chapter, we describe the implementation of the exchange platform.

Chapter 5

Implementation

In the previous chapter, we discussed the elements of the exchange platform, described the possible implementations, considered their advantages and disadvantages, and picked a solution. Here we describe the implementation of the exchange platform.

5.1 Workflow

Here the intended workflow of the exchange platform is described, from creating the lab exercise, creating its Git project, to portal access. The current implementation is made with *labsystem* integration in mind. Due to time and scope constraints, some *labsystem* integration are only discussed but not fully implemented.

The *labsystem*'s role in the system is to create and perform lab exercises. After a lab exercise is created in the *labsystem*, a script is run to push the lab contents to the Git repository. Three projects are created: lab project containing lab files, solutions project containing its solutions, and feedback project to store feedback result. Once it is stored in the Git repository, it is accessible from the frontend portal. There, user can use the portal website to view the lab information and give feedback.

The script to create and update lab from the *labsystem* and GitLab is discussed separately in Section 5.5. The portal features are discussed in Section 5.3. Before we discuss them, we describe the GitLab implementation as discussed in previous chapter.

5.2 GitLab

As mentioned in Section 4.3, we use GitLab to serve as the backend of the exchange platform. In this section we describe the GitLab implementation discussed in Chapter 4.

5.2.1 Setup and Installation

For our implementation, a virtual machine is created to serve as deliverable artifact. The specs of the virtual machine are:

```
OS: Debian GNU/Linux 9.0 (stretch) x86_64
Kernel: 4.9.0-3-amd64
CPU: Intel Xeon E5-1650 v3 (2) @ 3.5GHz
Memory: 2356MB / 4004MB
```

GitLab can be installed via multiple installation methods [68]. On the virtual machine, the installation is done with Omnibus package installation via the following commands.

```
#Install and configure the necessary dependencies
sudo apt-get install curl openssh-server ca-certificates
    postfix

#Add the GitLab package server and install the package
curl -sS https://packages.gitlab.com/install/repositories/
    gitlab/gitlab-ce/script.deb.sh | sudo bash
sudo apt-get install gitlab-ce

#Configure and start GitLab
sudo gitlab-ctl reconfigure
```

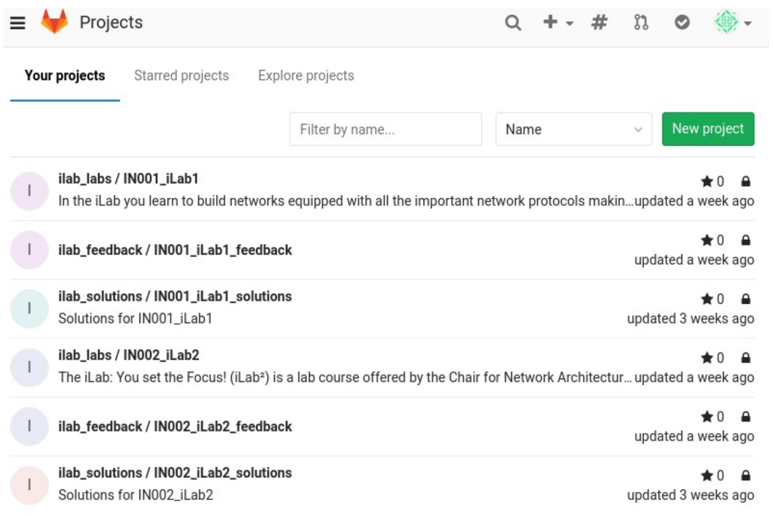
GitLab configurations can be found in `/etc/gitlab/gitlab.rb` [69]. In our setup, we use the default settings. The GitLab installation can be reached (via web interface) in `ilabxp2.net.in.tum.de:9080`.

For our implementation, we use the latest (at the time of writing) version of GitLab, `9.3`. We use GitLab Community Edition since their available features are sufficient for our requirements. GitLab Enterprise Edition could be considered for its LDAP integration features.

5.2.2 Structure

In this setup, we assume that a GitLab account is available for each institution or department, which is responsible for administration purposes. The rest of the writing also assumes that this account creates, edits, and administers the Git projects.

Groups are made to separate labs, solutions, and feedback. In this implementation, they are named *ilab_labs* for labs, *ilab_solutions* for solutions, and *ilab_feedback* for feedback. GitLab version `9.3` offers subgroups to further divide them. For our purposes, this is not necessary. The access control of these groups are set to members only. Later, accounts requesting access to the labs or its solutions should be granted access to the groups, rather than the individual projects.



ilab_labs / IN001_iLab1
In the iLab you learn to build networks equipped with all the important network protocols makin... updated a week ago

ilab_feedback / IN001_iLab1_feedback
updated a week ago

ilab_solutions / IN001_iLab1_solutions
Solutions for IN001_iLab1
updated 3 weeks ago

ilab_labs / IN002_iLab2
The iLab: You set the Focus! (iLab²) is a lab course offered by the Chair for Network Architectur... updated a week ago

ilab_feedback / IN002_iLab2_feedback
updated a week ago

ilab_solutions / IN002_iLab2_solutions
Solutions for IN002_iLab2
updated 3 weeks ago

Figure 5.1: Labs, solutions, and feedback as projects, hosted in GitLab

Each labs and their solutions are created as separate Git projects. Each of these projects contains all the files of that lab. Lab files created by the *labsystem* creates a subfolder containing the solutions. This must be handled separately, and is discussed further in Section 5.5. The main project containing the lab is named `labname`, while the project containing the solutions is named `labname_solutions`

A feedback project is also created to store feedback, named `labname_feedback`. The primary storage of survey data is a database, and this Git project is not updated each time a survey is submitted. The database implementation is described in Section 5.4. The main purpose of this approach is to store comments and backup of survey data.

GitLab *issue* is created to host the lab's comments. Each comments from the frontend portal are posted as comments to this project's issue. The issue is named `labname_comments` in project `labname_feedback`.

While the survey data is stored in a separate database, an additional backup is stored in the Git repository as well. This is made in case the survey data needs to be analyzed separately outside the exchange platform. The survey data in the Git project is stored in *json* format, in a file named `survey.json`.

Listing 5.1: Survey data stored in JSON format

```
[
  {
    "id": 201,
    "s1": 4,
    "s2": 4,
    "s3": 3,
    "s4": 8,
    "s5": 19,
    "rating": 5,
    "comment": "I am a comment",
    "sha": "34cca73a",
    "branch": "master",
    "project_id": 1,
    "date": "2017-01-01 00:00:00"
  },
]
```

This *json* file is only updated when the export function is used from the frontend portal. To save space, this *json* is also minified (not shown above). In the current implementation, only the *master* branch is used. In case of storing multiple data (for example from multiple institution), it is not sufficient. However, this is not a concern for our current implementation and this is left for future improvements.

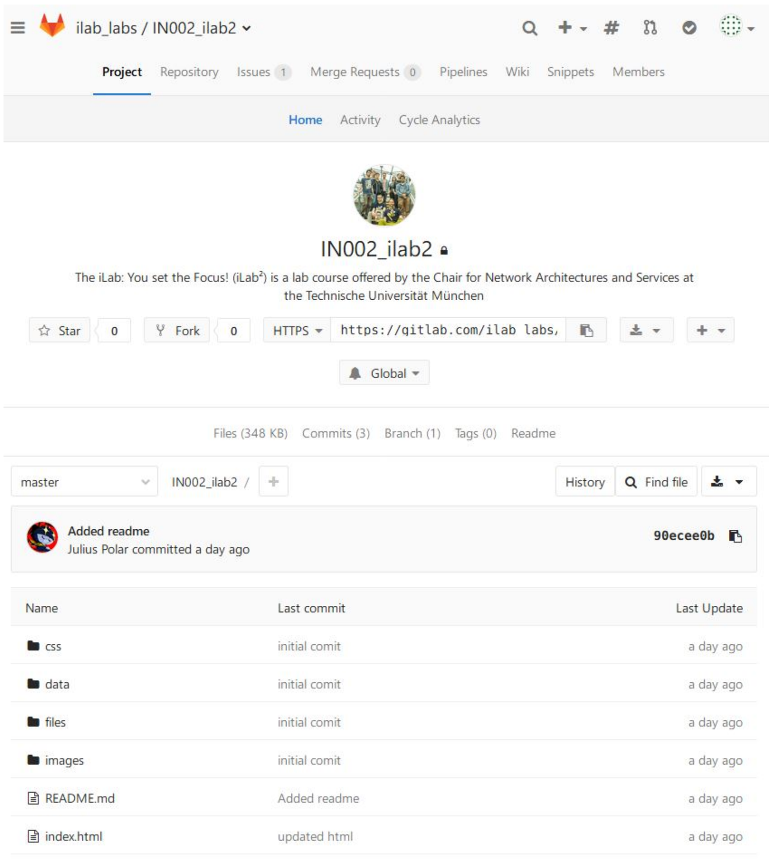
As mentioned in Chapter 4, the metadata of the lab can be stored in multiple ways. Here the description that we want to show in the portal is saved in the project's `description`. Additionally, label(s) can be added to the project to serve as filter tags in the frontend. These description and labels can be added in either the import script or from GitLab web interface.

A GitLab account is made for the portal, in order to specify what the portal has access to. This account is given access to the *ilab_labs* and *ilab_feedback* group. The level of access is *Guest*, since it does not need access to repository files. In case of integration with already existing repository, this account is still needed, and need to be given access to the lab groups with at least *Guest* permissions.

For the sake of demonstration, a teacher account is also made. This account has access to all groups mentioned above, therefore having access to the solutions. The level of access is *Developer*, since it requires access to the project files. In case of integration with already existing repository, anyone requiring access to the solutions should be given at least *Developer* permissions.

5.3 Frontend Portal

Now that we have our GitLab setup and installed, let's look at the portal.



IN002_ilab2

The iLab: You set the Focus! (iLab²) is a lab course offered by the Chair for Network Architectures and Services at the Technische Universität München

Added readme
Julius Polar committed a day ago
90ecee0b

Name	Last commit	Last Update
css	initial comit	a day ago
data	initial comit	a day ago
files	initial comit	a day ago
images	initial comit	a day ago
README.md	Added readme	a day ago
index.html	updated html	a day ago

Figure 5.2: Lab project in GitLab

5.3.1 Setup and Installation

The frontend is written in HTML, CSS, and Javascript, using AngularJS framework. The frontend source code is available in

`https://gitlab.com/JulianElda/Exchange_Platform_Frontend`

The frontend can be installed by pulling the code from the repository and serving the directory with a web server. Apache2 is used as webserver to host the frontend. The portal is accessible via `ilabxp2.net.in.tum.de`. In the virtual machine, the frontend can be enabled/disabled by `a2ensite frontend.conf`.

Listing 5.2: Frontend site configuration file for apache2

```
<VirtualHost *:80>
  DocumentRoot /var/www/Exchange_Platform_Frontend
  ServerName ilabxp2.net.in.tum.de

  <Directory />
```

```
    Allow from all
    AllowOverride All
    Options FollowSymlinks
    Require all granted
  </Directory>
</VirtualHost>
```

Frontend configurations are specified in `config.json`.

Listing 5.3: Frontend configuration file

```
{
  "git_host": "http://ilabxp2.net.in.tum.de:9080",
  "git_api": "api/v4",
  "db_host": "http://ilabxp2.net.in.tum.de",
  "portal_token": "3kQUqRuDGfmrr51JqYj8",
  "lab_preview_path_prefix": "preview",
  "labs_group_id": 3,
  "solutions_group_id": 4,
  "feedback_group_id": 6
}
```

The configuration attributes are (some options are omitted):

- `git_host`: hostname or address of the GitLab repository, with port number as needed
- `git_api`: GitLab api version.
- `db_host`: hostname or address of the survey database.
- `portal_token`: private token of the frontend GitLab account.
- `lab_preview_path_prefix`: directory containing the labs in the server. Needed for lab preview.
- `labs_group_id`: GitLab group id of *ilab_labs* group containing lab files.
- `solutions_group_id`: GitLab group id of *ilab_solutions* group containing lab solutions.
- `feedback_group_id`: GitLab group id of *ilab_feedback* group containing lab feedback.

5.3.2 Backend Communication

As previously discussed, the portal communicates with GitLab repository via GitLab API. In this section we detail how the implementation is done.

GitLab API endpoints can be accessed by appending `api/api_version` to GitLab hostname. In this implementation, api version **4** is used. For example, getting the list of users can be accessed via `GET /users`. The full url of the (GET) request is

```
ilabxp2.net.in.tum.de:9080/api/v4/users
```

For brevity, the short notation (method and url) i.e. `GET /users` is used for writing, without the full hostname and api version.

So far, the projects and groups described are private. Accessing a private resource without authentication results in `401 Unauthorized`. Since a frontend portal account is created, the requests made by the portal website is made such that the user is authenticated with the portal account. This is done by using *Private Token*. Private tokens provide full access to the GitLab API. Anyone with access to them can interact with GitLab as if they were authenticated as that token owner. The token can be found in the GitLab account page account page (`/profile/account`) after logging in to GitLab web interface. This token can be appended into the request url as a parameter

```
ilabxp2.net.in.tum.de:9080/api/v4/users?private_token=123abcdef
```

or passed as header in the request. For example, a request made with `curl`:

```
curl -header "PRIVATE-TOKEN: 123abcdef" "ilabxp2.net.in.tum.de:9080/api/v4/users"
```

The private token of the frontend portal is then added to either the url or the header of the request. In this implementation it is appended to the headers to keep the request url short. After attaching the private token, all requests are made as if the frontend portal is logged in to GitLab.

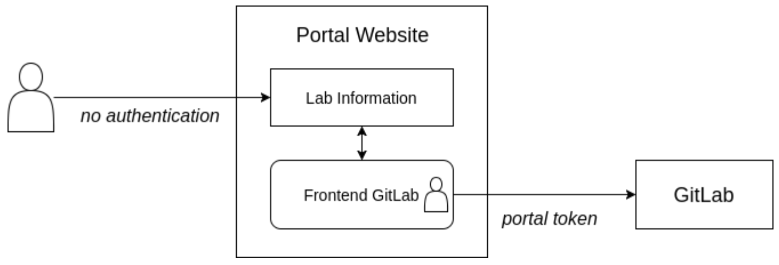


Figure 5.3: Private Token authentication with GitLab API

In addition, an authentication feature is also implemented in the portal to login with

GitLab account (from the same GitLab repository) using GitLab API. All the requests on the frontend then uses this token instead of portal token. This way the portal can serve a different content for authenticated user, and provide access of the user's projects that the user has access to. After authentication, an account with access to the solutions can see them in the lab description. The activity feed is also updated with the user's activity feed. Lab commenting is also made available.

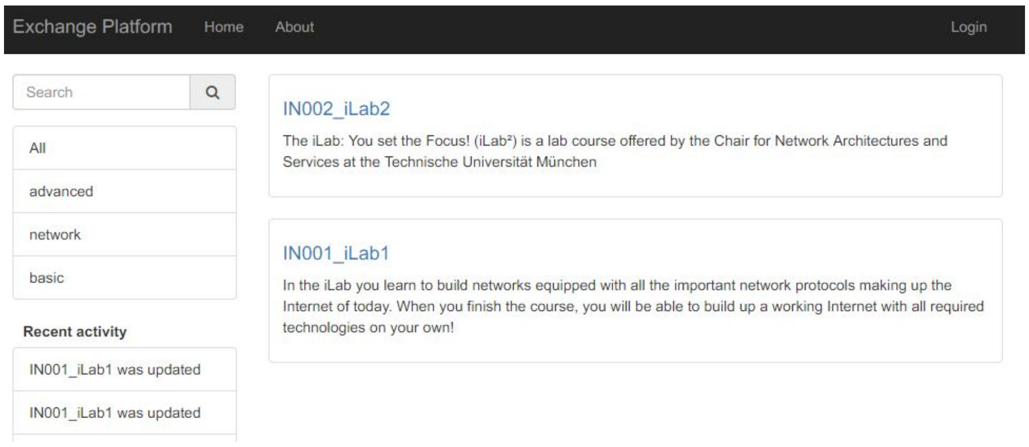
The main purpose of authentication is to access solutions. After authenticating with users with solutions access, they are accessible in the lab's description page. The other feature affected by logging in is activity feed, where it uses the authenticated user's activity feed instead of the frontend portal's.

5.3.3 Portal Components

The portal has three main components: catalog, description, and survey. The catalog contains the list of labs, description contains the lab's description, preview, and comments, while the survey component contains survey submission, token, and result.

5.3.4 Portal Catalog

The catalog serves as the main page of the website. Its primary function is to show the list of labs in the repository.



IN002_iLab2

The iLab: You set the Focus! (iLab²) is a lab course offered by the Chair for Network Architectures and Services at the Technische Universität München

IN001_iLab1

In the iLab you learn to build networks equipped with all the important network protocols making up the Internet of today. When you finish the course, you will be able to build up a working Internet with all required technologies on your own!

IN001_iLab1 was updated

IN001_iLab1 was updated

Figure 5.4: Exchange Platform Frontend Lab Catalog

The list of labs can be seen in the middle of the page. The attributes shown are lab title and description. A search bar and tag filter is provided on the left side to search for lab. Below the search bar is an activity feed of the authenticated user (default frontend portal).

As mentioned in Section 4.4, using `GET /projects` returns all projects, including projects unrelated to the frontend. The projects we want to show here is under the group *ilab_labs*. This is why the `id` of the lab group is stored in the configuration file, so that we can use `GET /groups/:group_id/projects`, to get the list of lab projects.

Searching through the catalog is done in two ways: using the search bar and/or tag filter. The search bar filters through the lab's name and description as the user type in it. The tag filter filters through the lab project's `tag_list`. The tags of the project is specified in the creation of the project, and can later be modified via GitLab web interface in the project settings. The two can be used in combination, e.g. search keyword *iLab* under the tag *network*.

The activity feed is inspired by activity feed in GitLab. GitLab web interface provides a list of latest activities in the projects the user has access to. Unfortunately, the functionality offered by GitLab API only returns the activities of the authenticated user (instead of activities from all users), and only on one specific project, meaning creating an activity list of all projects requires looping through the project list and queries their activity list, which is the current implementation in the portal. Regardless, the API returns activities such as commits, merge requests, issues, comments, or administration changes. The frontend only parses new comments, new commit events, and new projects. The current implementation still leaves a lot for improvements, and might change on the development of GitLab's APIs.

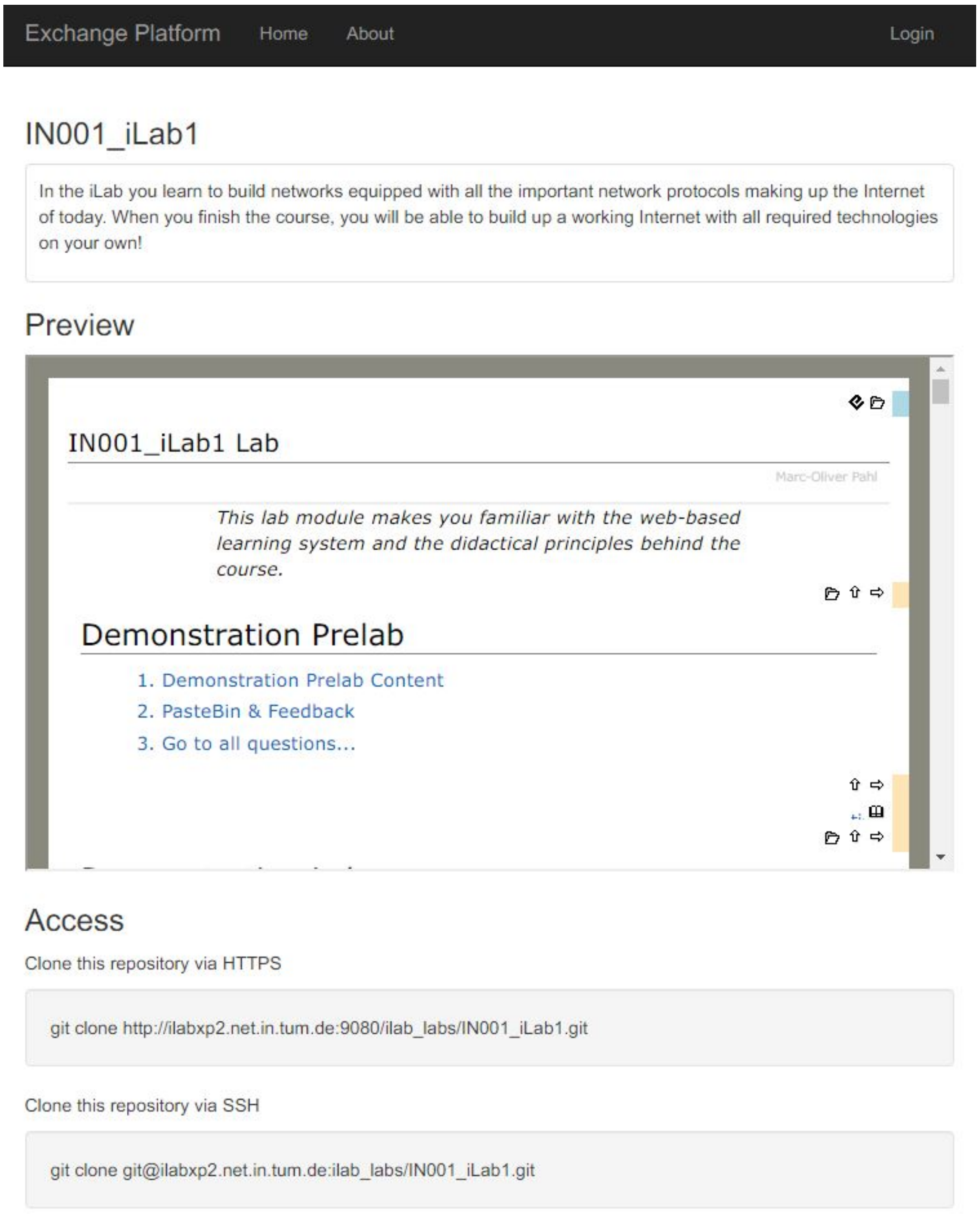
5.3.5 Lab Description

Clicking on the title of the lab from the catalog opens up the lab's description. Shown in the page is the lab title, description, preview, and comments. The lab information is obtained via `GET /projects/:project_id` endpoint. The lab's title and description is taken from the project's `name` and `description` respectively.

A feature unique to the portal is lab preview. This is due to the HTML format of labs generated by the *labsystem*. An *iframe* is created to show the generated HTML in the page. However, this implementation requires the frontend server to have a copy of each lab for the *iframe* to work. In the current implementation, the labs are contained in a folder named `preview` in the frontend website. This is configurable in the frontend configuration file (`config.json`).

Git repository address of that project is shown as well, in HTTP(S) and SSH. A direct download-project as archive feature was previously considered but not implemented. This is to encourage use of Git to contribute to the project, since any interested user has to request access to the repository and interact with the system, compared to directly getting the zipped archive where the user does not have to.

If the user is authenticated, and has access to the solutions group (*iLab_solutions*), a


Exchange Platform Home About Login

IN001_iLab1

In the iLab you learn to build networks equipped with all the important network protocols making up the Internet of today. When you finish the course, you will be able to build up a working Internet with all required technologies on your own!

Preview

IN001_iLab1 Lab

Marc-Oliver Pahl

This lab module makes you familiar with the web-based learning system and the didactical principles behind the course.

Demonstration Prelab

1. Demonstration Prelab Content
2. PasteBin & Feedback
3. Go to all questions...

Access

Clone this repository via HTTPS

```
git clone http://ilabxp2.net.in.tum.de:9080/ilab_labs/IN001_iLab1.git
```

Clone this repository via SSH

```
git clone git@ilabxp2.net.in.tum.de:ilab_labs/IN001_iLab1.git
```

Figure 5.5: Lab page showing lab description, preview, and Git access

button to access the solution is shown in this page as well. Clicking on the button brings the user to a similar page to the lab description page. The solutions page shows the title (always named *labname_solutions*), description, and Git repository access.

So far, the frontend portal shows lab information from GitLab, and information flows one way from GitLab to the portal (except authentication). Another major component of the portal is feedback. In the next section we discuss the implementation of feedback components in the portal.

5.3.6 Lab Feedback

Feedback is another major component on the portal. As discussed in Chapter 4, we choose to implement reviews, surveys, and ratings. In the portal, these are implemented in two major features: comments and surveys.

As discussed in Section 4.1.6.1, review and rating are combined in one survey. Here are the survey question we implement:

- 1. How difficult was the lab for you? (1) Easy - (5) Difficult
- 2. How interesting was the lab for you? (1) Boring - (5) Interesting
- 3. How long was the lab for you? (1) Short - (5) Long
- 4. How much time did you spend on the pre-lab?
- 5. How much time did you spend on the lab?
- 6. How would you rate this lab?
- 7. Do you have any particular feedback? Give us your comments

The survey workflow begins from the *labsystem*: the student is given a token (one for each lab) required to submit a survey. The student is given a url to the the survey page of the lab in the frontend portal, with the token as a GET query parameter (so that the student doesn't have to type it in the portal). Additional parameters required are the (GitLab) project id of the lab, name of the Git branch of the lab, and the commit id of the lab project. The student then fills in the survey form and submit it.

Once submitted, the survey data is saved in a database. The database is discussed later in Section 5.4. In addition to being saved in the database, the survey data can be exported to the GitLab repository. This feature is accessed from the lab information page in the portal. Survey export and token generation requires authentication. In the survey export page, the user can select to export the survey data to that lab's feedback project under *iLab_feedback* group. A specific branch and/or commit id of the project (of the survey data) can be selected, by default this feature exports all survey data of the lab. Once initiated, the survey data is saved in a *json* format in *labname_feedback* project. A download from browser feature is also provided to download the *json* file directly from the browser. This stored *json* is not meant to serve as the primary storage of the survey data, but as secondary backup data. The survey result page obtains the data from the database, not from backup file in GitLab.

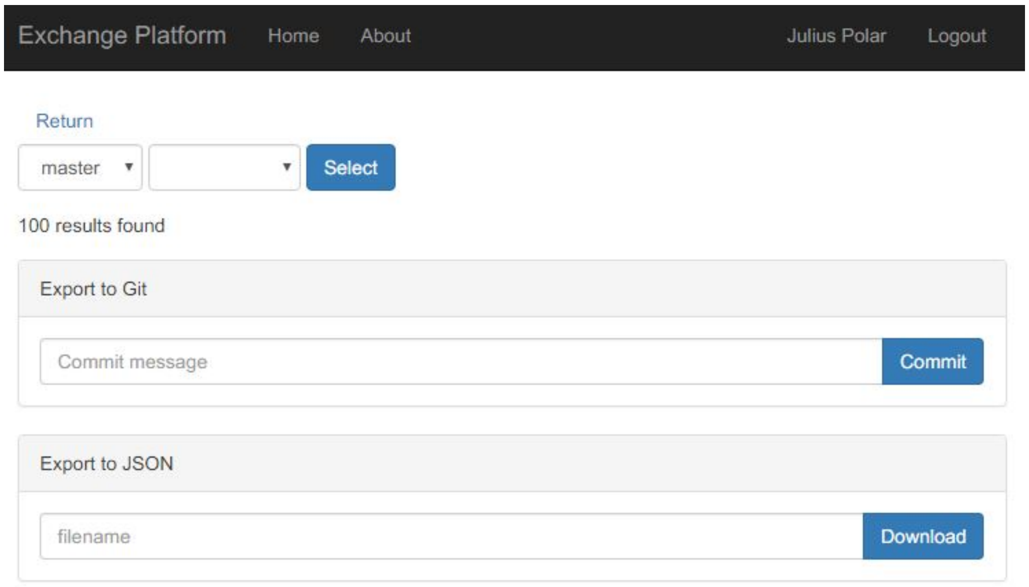


Figure 5.6: Exporting survey data

The result of the survey can be accessed without authentication from the lab information page. In the survey result page, the survey data is analyzed and visualized. On the top, Git branch and commit id of the lab can be selected to filter survey result of that branch. The resulting visualization is grouped into three parts in the frontend.

The first part shows a horizontal stacked bar chart, combined for questions number 1, 2, 3, and 6. The bar chart shows a proportional ratio of each answers compared to the total number of submitted answers. A more detailed view is provided below: a select-dropdown is provided to choose a question, and a separate chart is shown. For this chart, another select-dropdown is provided to view it in either a pie or a bar chart. Along with the chart, the average, median, minimum and maximum value of the data is shown.

The second part shows a histogram each for both question number 4 and 5. The x-axis shows the submitted answers (*n* amount of hours), while the y-axis shows the number of occurrence of the x-axis value. Similar to the first part, the average, median, minimum, maximum, and standard deviation of the data is also shown.

The third part shows the submitted answers from question number 7. Since this question is not mandatory, i.e. the survey can be submitted without answering this question, empty values are not shown. To prevent excessive scrolling, this list is paginated, showing 10 answers each.

The survey is the main method of gathering feedback for the lab. There is one more

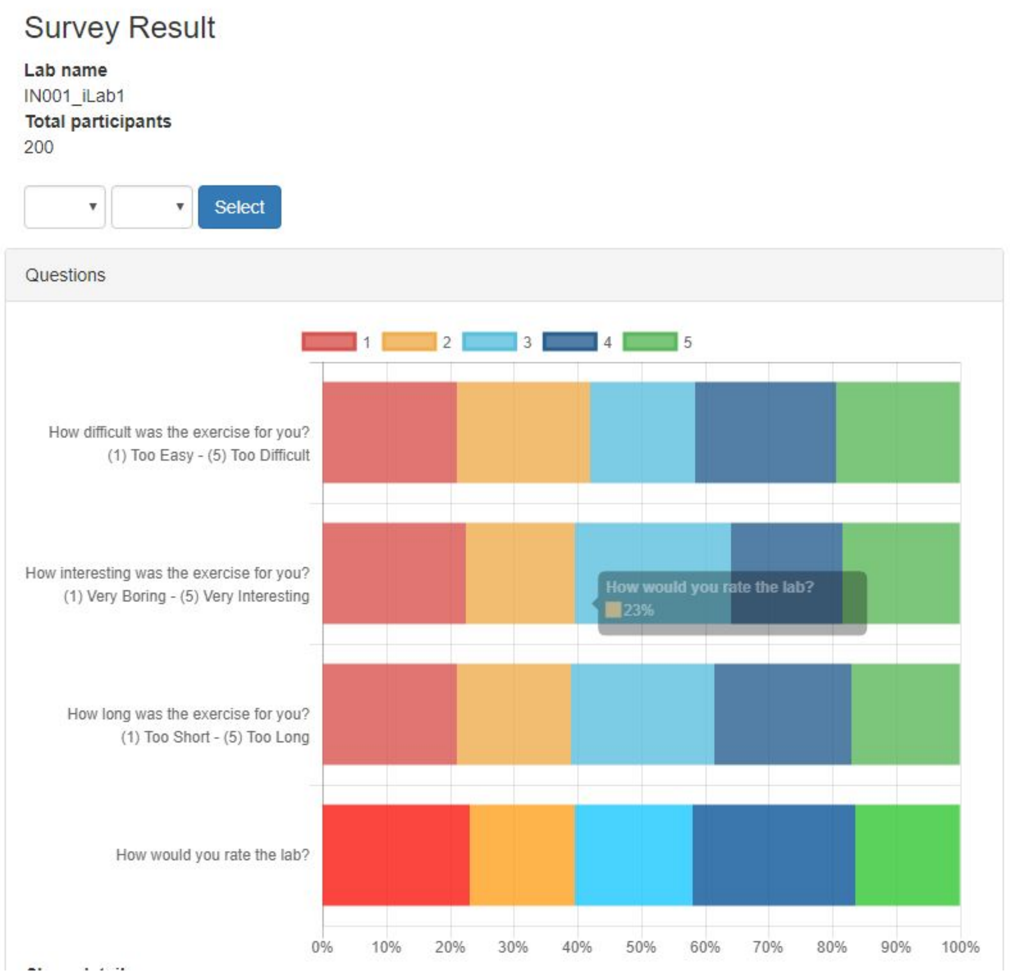
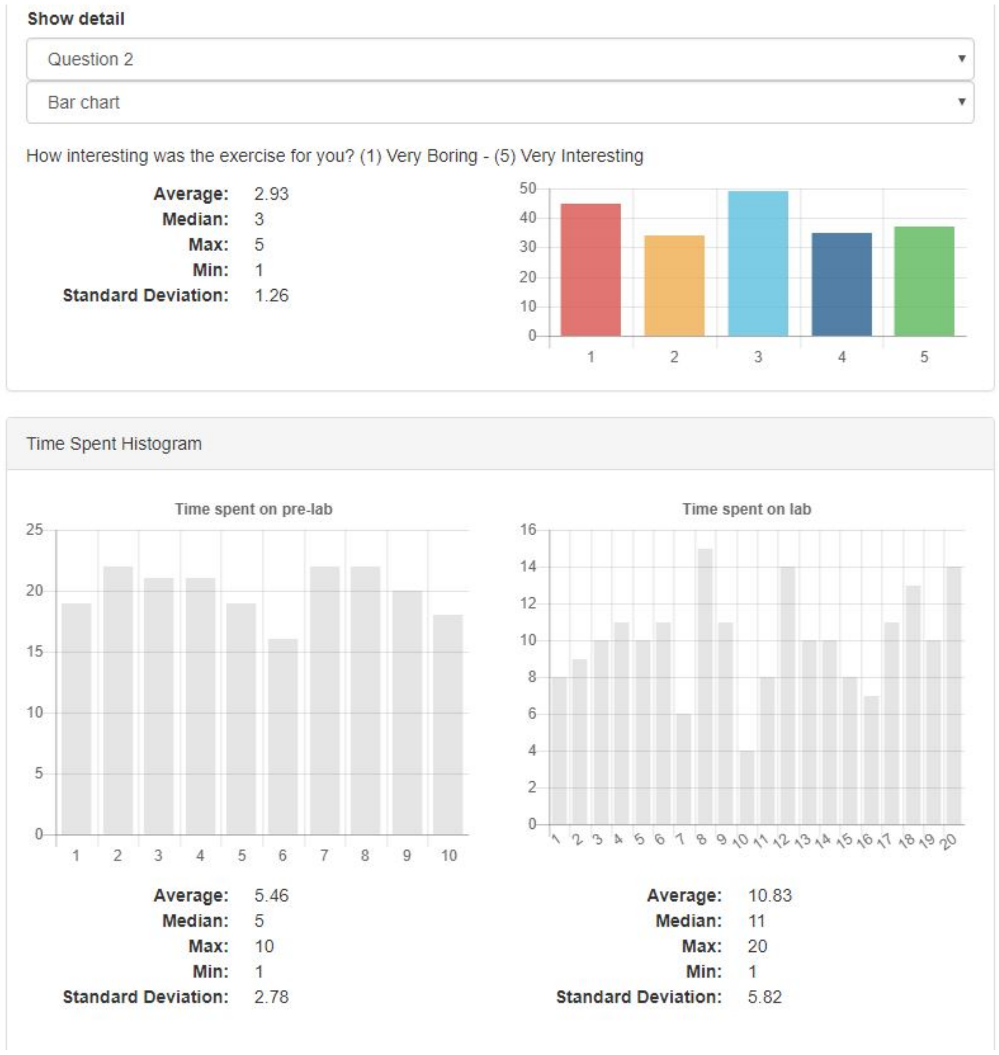

Survey Result
Lab name
IN001_iLab1
Total participants
200
Select
Questions
1
2
3
4
5
How difficult was the exercise for you? (1) Too Easy - (5) Too Difficult
How interesting was the exercise for you? (1) Very Boring - (5) Very Interesting
How would you rate the lab?
23%
How long was the exercise for you? (1) Too Short - (5) Too Long
How would you rate the lab?
0%
10%
20%
30%
40%
50%
60%
70%
80%
90%
100%

Figure 5.7: Survey result visualization for question 1, 2, 3, and 6

feedback method implemented in the portal: comments. The comment feature is not combined with the survey mentioned before, but uses GitLab comments. During research and development of this thesis, this GitLab comment was the initial feedback gathering feature, before it was replaced by survey. However, it still has some advantages and we decided to keep the feature.

This comment feature uses GitLab issue and comment feature. An issue is created for each lab project to contain the comments. The issue is named *labname_comments* and labeled as *comments* to separate it from other issues (in case the GitLab issue feature is needed for issue tracking). This issue is not created under the lab project in the lab group (*iLab_labs*), but in the feedback project in the feedback group (*iLab_feedback*).

The list of the project issue is obtained via `GET /projects/:project_id/issues`. After filtering it to get the issue assigned for comments, the id of the issue is used to get



How interesting was the exercise for you? (1) Very Boring - (5) Very Interesting

Average: 2.93
Median: 3
Max: 5
Min: 1
Standard Deviation: 1.26

Time spent on pre-lab

Average: 5.46
Median: 5
Max: 10
Min: 1
Standard Deviation: 2.78

Time spent on lab

Average: 10.83
Median: 11
Max: 20
Min: 1
Standard Deviation: 5.82

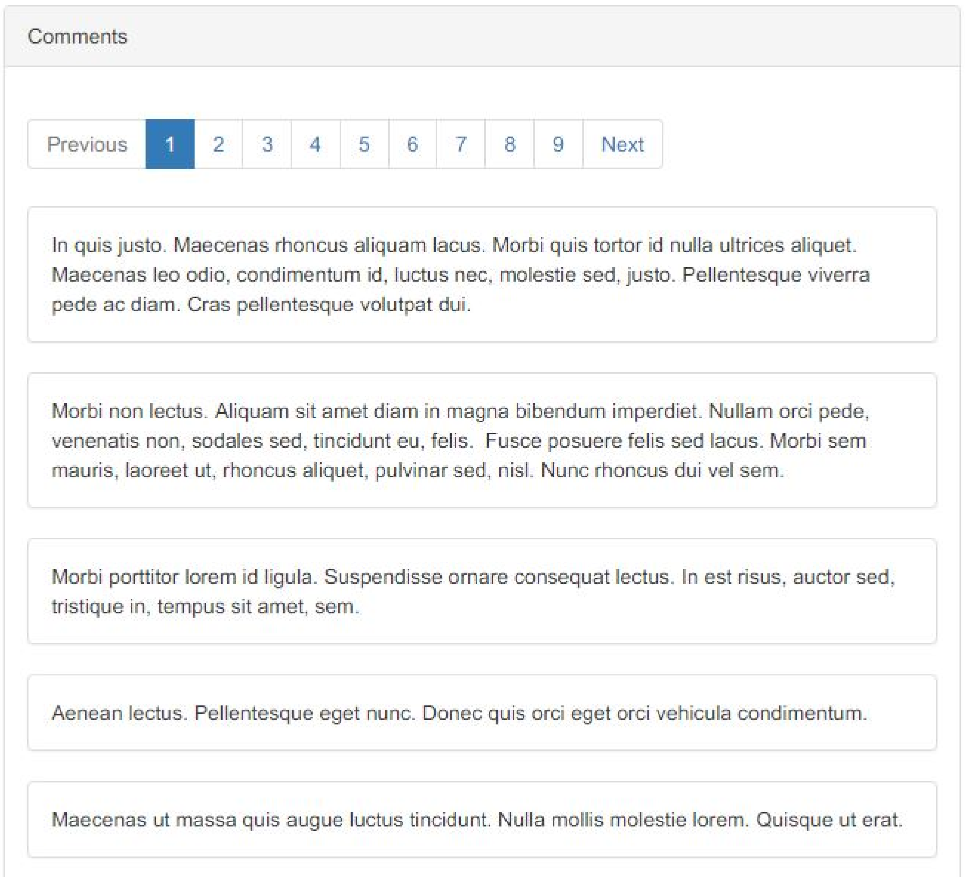
Figure 5.8: Detailed chart for question number 1, 2, 3, and 6, and histogram for question number 4 and 5

the comments of the issue via `GET /projects/:project_id/issues/:issue_id/notes`. The results are then parsed and shown in the bottom of the page, sorted latest first. A textfield is provided to create a comment on the issue via

```
POST /projects/:project_id/issues/:issue_id/notes
```

Authentication is required to comment on labs to prevent spam. Once submitted on the portal, a comment on the issue in GitLab is made, posted by the authenticated user.

This commenting feature has some disadvantages. It requires authentication, so a student wanting to comment needs to login, and needs a GitLab credentials to be created, compared to the survey that uses generated token. And since it is a GitLab comment,



In quis justo. Maecenas rhoncus aliquam lacus. Morbi quis tortor id nulla ultrices aliquet. Maecenas leo odio, condimentum id, luctus nec, molestie sed, justo. Pellentesque viverra pede ac diam. Cras pellentesque volutpat dui.

Morbi non lectus. Aliquam sit amet diam in magna bibendum imperdiet. Nullam orci pede, venenatis non, sodales sed, tincidunt eu, felis. Fusce posuere felis sed lacus. Morbi sem mauris, laoreet ut, rhoncus aliquet, pulvinar sed, nisl. Nunc rhoncus dui vel sem.

Morbi porttitor lorem id ligula. Suspendisse ornare consequat lectus. In est risus, auctor sed, tristique in, tempus sit amet, sem.

Aenean lectus. Pellentesque eget nunc. Donec quis orci eget orci vehicula condimentum.

Maecenas ut massa quis augue luctus tincidunt. Nulla mollis molestie lorem. Quisque ut erat.

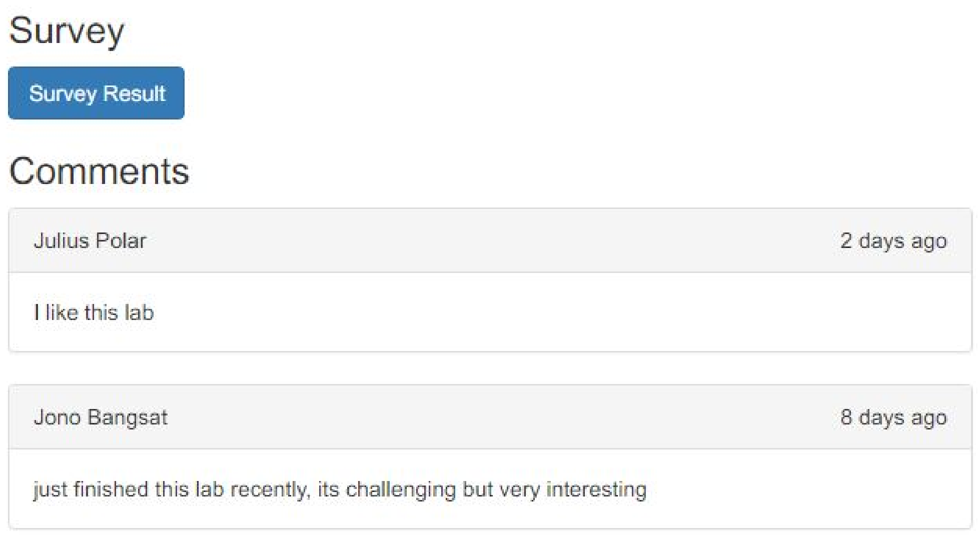
Figure 5.9: Survey result visualization for question number 7

it is stored in GitLab, so any export action requires accessing both the database and GitLab. Regardless, it still have some good use, such as used by teachers to post news or information about the lab. For example, a teacher can comment about the lab questions being corrected, and anyone visiting the page can see the comment.

To summarize, the portal shows a catalog list, activity feed, lab description, preview, survey, and comments. The information shown on the catalog are all obtained via GitLab APIs. Further improvements is discussed in Chapter 7. In the next section, we describe the database implementation for storing survey data.

5.4 Database

In the exchange platform, the database is used for two purposes: storing survey token, and survey data. In this section we describe the structure of the database and its tables.



Survey

Survey Result

Comments

Julius Polar 2 days ago

I like this lab

Jono Bangsat 8 days ago

just finished this lab recently, its challenging but very interesting

Figure 5.10: Lab page showing survey result and comments

5.4.1 Setup and Installation

For our implementation, MySQL version `15.1 Distrib 10.1.23-MariaDB` is used. Additionally, *phpmyadmin* is also installed to access MySQL with a web interface. The two can be installed via different methods; for this implementation we installed them from Debian's package manager. The database is installed in the virtual machine with the following commands:

```
#Install and configure MySQL and phpmyadmin
sudo apt install mariadb-server phpmyadmin
```

The database can be accessed (using *phpmyadmin*) in `ilabxp2.net.in.tum.de/phpmyadmin`. Once the database is installed, we can populate them with the survey data and token. But before that, we need to describe the database and table structure.

5.4.2 Structure

The exchange platform needs one database to store its data. We create a database named `exchange_platform` for this purpose. A specific user for this database is also created, named `ilabxp`.

There are two tables in the `exchange_platform` database: `survey_token` and `survey_result`. There are no relationship between the two table.

```
MariaDB [exchange_platform]> describe survey_result;
+------------+---------------+-------+-----+------------------+---------------------------+
| Field      | Type          | Null | Key | Default           | Extra                       |
+------------+---------------+------+-----+-------------------+-----------------------------+
| id         | int(11)       | NO   | PRI | NULL              | auto_increment              |
| s1         | int(11)       | NO   |     | NULL              |                             |
| s2         | int(11)       | NO   |     | NULL              |                             |
| s3         | int(11)       | NO   |     | NULL              |                             |
| s4         | int(11)       | NO   |     | NULL              |                             |
| s5         | int(11)       | NO   |     | NULL              |                             |
| rating     | int(11)       | NO   |     | NULL              |                             |
| comment    | varchar(1024) | YES  |     | NULL              |                             |
| sha        | varchar(11)   | NO   |     | NULL              |                             |
| branch     | varchar(64)   | NO   |     | NULL              |                             |
| project_id | int(11)       | NO   |     | NULL              |                             |
| date       | datetime      | YES  |     | CURRENT_TIMESTAMP | on update CURRENT_TIMESTAMP |
+------------+---------------+------+-----+-------------------+-----------------------------+
```

The table `survey_result` is used to store survey result. The answer of survey questions from Section 5.3.6 are stored here. Questions 1 to 5 from the survey are stored in `s1` to `s5`, while question 6 is stored as `rating`, and question 7 as `comment`. The lab identifier is stored as `project_id` since it refers to its GitLab project id. To keep track of the lab version of the survey, the branch name and commit id are stored as `branch` and `sha` respectively. Additionally, the `id` is used as unique identifier of the entry in the database, while `date` is used as a timestamp of when the survey is submitted. All fields other than `id`, `comment`, and `date` are mandatory.

```
MariaDB [exchange_platform]> describe survey_token;
+-----------------+-------------+------+-----+---------+----------------+
| Field           | Type        | Null | Key | Default | Extra          |
+-----------------+-------------+------+-----+---------+----------------+
| id              | int(11)     | NO   | PRI | NULL    | auto_increment |
| token           | varchar(64) | NO   |     | NULL    |                |
| expiration_date | datetime    | NO   |     | NULL    |                |
| project_id      | int(11)     | NO   |     | NULL    |                |
+-----------------+-------------+------+-----+---------+----------------+
```

The table `survey_token` is used to store survey tokens. The `id` field serves as unique identifier of the database row, `token` stores the survey token, `expiration_date` is the expiration date of the token, while `project_id` stores the lab which the token belongs to.

A library named *PHP-CRUD-API* is used to send queries to the database from the portal. The library can be found in `https://github.com/mevdschee/php-crud-api`.

Now that we have described the GitLab, frontend portal, and database structure and implementation, we can close by describing how we integrate the *labsystem* with the system we have made so far.

5.5 Labsystem integration

The role of the *labsystem* in this exchange platform is twofold: creating and updating lab contents (and solutions) to the GitLab, and providing survey link to the portal.

The survey link is very straightforward: the survey component of the portal is accessed by a url, with some parameters attached as GET query parameters. This request should point to the survey module in the portal: `/survey`. The parameters required are as follows:

- `id`: GitLab project id of the lab
- `token`: survey token
- `branch`: branch name of the GitLab project of the lab
- `sha`: commit id of the lab version

All of the parameters are required. The URL can be obtained by logging in to the portal and accessing the tokens page of the lab. The portal parses the given parameters, and show a confirmation before starting the survey. An example of a survey url would be:

```
ilabxp2.net.in.tum.de/#!/survey?id=12&
   token=123abcdef&branch=master&sha=abcdef
```

The other role of the *labsystem* is pushing lab contents to the GitLab repository. This covers two scenarios: pushing new labs, and pushing updates to existing labs. Both are done with a shell script provided in the source code (`new.sh` and `update.sh`). In the current implementation, these are not directly integrated in the import/export feature in the *labsystem* and must be executed manually from a shell. The variables used by the script can be changed in `config.cfg`.

The export script (`export.sh`) requires two parameters: the project name to be created in GitLab, and path to the lab folder directory. Here is a breakdown of the script to push a new lab to the GitLab repository:

- change working directory to the lab folder directory.
- create a new project containing the lab files under *iLab_labs* group in GitLab using GitLab API. The API is called using *curl*.
- initialize a new Git repository in the working directory using `git init`.
- set the remote url to the url of the newly created git project. The url is parsed from *curl* reply from previous step.
- add the solutions directory to `.gitignore`.
- add all items to git index using `git add -all`.
- create an initial commit using `git commit -m "initial commit"`.
- push the files to GitLab using `git push -u origin master`.
- create a new project for feedback under *iLab_feedback* group in GitLab using GitLab API.

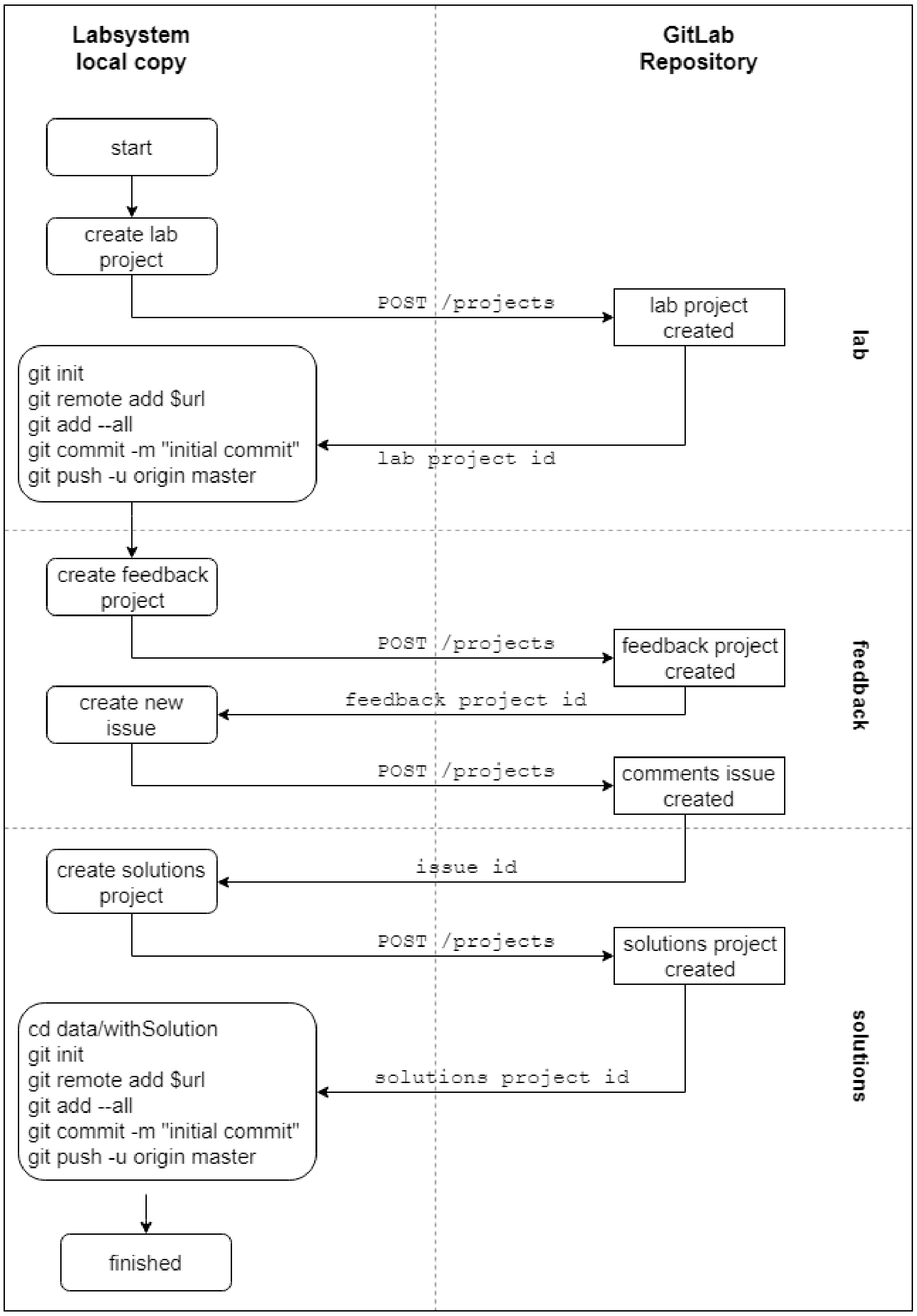


Figure 5.11: Export script flow chart

- create a new issue named `labname_comments` in the feedback project for comments. This is also done using GitLab API.
- change working directory to the lab solutions directory.
- create a new project containing the lab solutions under *iLab_solutions* group in GitLab using GitLab API.
- add all items to git index using `git add -all`.
- create an initial commit using `git commit -m "initial commit"`.
- push the files to GitLab using `git push -u origin master`.

The other script, `update.sh` is used to push the changes in the local copy in the *labsystem* installation to the GitLab repository. It requires two parameters: path to the lab folder directory and the commit message. Here is a breakdown of the update script:

- change working directory to the lab folder directory.
- print the changes in the git repository using `git status`.
- add all the changes with `git add -all`.
- commit the changes with `git commit -m "commit_message"`.
- push the changes using `git push -u`.
- get the latest commit id using `git log -n 1`.
- change working directory to the lab folder directory.
- change the latest commit id stored in a text file named `latest_commit` with the latest commit id
- print the changes in the git repository using `git status`.
- add all the changes with `git add -all`.
- commit the changes with `git commit -m "commit_message"`.
- push the changes using `git push -u`.

Additionally, an import script, `import.sh`, is also written to checkout labs from the GitLab. It requires two parameters: the GitLab project name of the lab, and directory to import the lab to. Here is a breakdown of the import script:

- change working directory to the specified path
- pull the lab project using `git clone`
- change working directory to `data`
- pull the solutions project using `git clone`

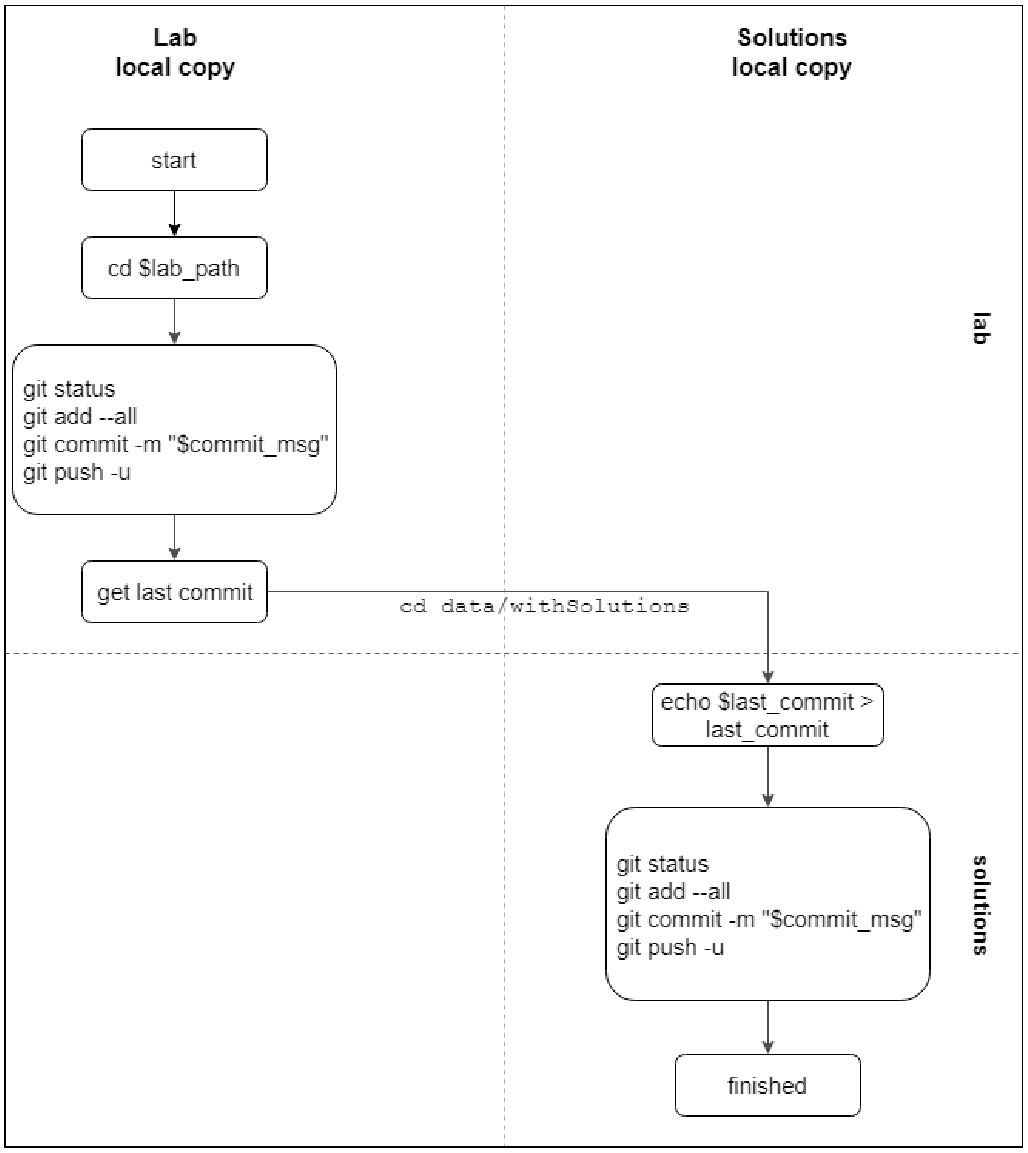


Figure 5.12: Update script flow chart

- renamed the solutions project directory name to `withSolutions` to keep it compatible with *labsystem* format

The *labsystem* scripts are the last component of our implementation. We have the *labsystem* to create and perform lab exercises, the *labsystem* scripts to import/export/update labs to the GitLab repository, a GitLab repository to store labs, a database to store survey token and data, and a frontend portal to show the labs and give lab feedback. In the next section, an example scenario is described that involves all the components of this implementation.

5.6 Example Scenario

The exchange platform we implement has many components, and to provide an overview of how the whole system interacts with one another, we described the following scenario. A teacher creates a new lab and wants to use the exchange platform to gather lab feedback via survey.

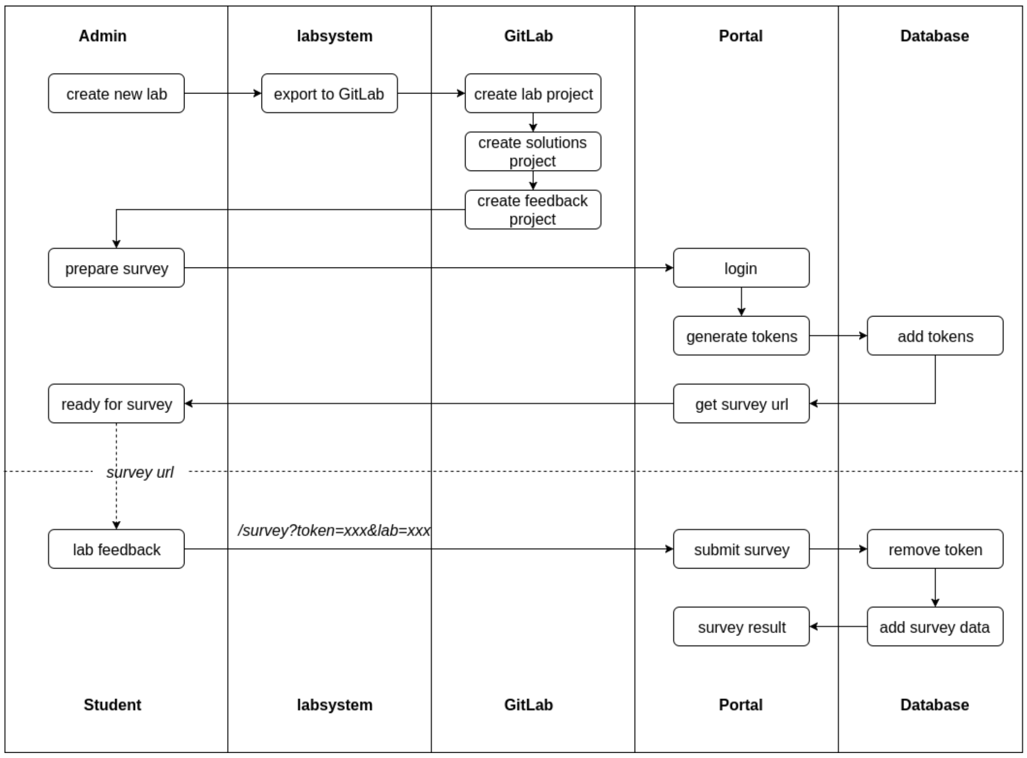


Figure 5.13: Sequence diagram of setting up lab for survey

A lab *IN001_iLab1* is created from the *labsystem*, and using the export script (`export.sh`), it is pushed to the GitLab repository. Three projects are created in the GitLab repository:

- *ilab_labs/IN001_iLab1*
- *ilab_solutions/IN001_iLab1_solutions*
- *ilab_feedback/IN001_iLab1_feedback*

The teacher heads to the frontend portal and login. The lab is selected from the catalog in the portal to show its description. The teacher then generate feedback tokens for lab survey. After specifying the branch and commit, URLs for the lab survey is generated. Each of this link is given to a student.

The student navigates to the given url and submit lab feedback. The token is deleted, and a survey data entry is added to the database. The lab survey result can then be seen in the frontend portal.

To conclude, we have implemented *labsystem* integration scripts to export/import labs, a GitLab repository to store labs and solutions, a database to store tokens and survey data, and a frontend portal to access and submit feedback for labs. The frontend source code and the integrations scripts are available in `https://gitlab.com/JulianElda/Exchange_Platform_Frontend`. In the next chapter, we compare the exchange platform with other related works and evaluate its advantages and disadvantages.

Chapter 6

Evaluation

In the previous chapters we described, analyzed, and implemented the exchange platform. In this chapter we conclude by assessing the exchange platform and comparing it with similar platforms.

6.1 Workflow Comparison

The evaluation is conducted on the following workflows:

- Giving lab feedback
- Viewing feedback result
- Browsing labs
- Creating and sharing labs
- Importing labs
- Versioning labs

The first three workflows are feedback related use cases, while the others are related to sharing labs. The first three workflows are compared with MOOC platforms for its feedback features, and the rest with Moodle and IEEE ComSoc Lab Exchange for their sharing features.

First we compare the feedback aspects of the exchange platform with MOOC platforms, due to their many tools provided to give feedback.

In the exchange platform, the lab feedback is given with a comment and a survey. The comment requires authentication, and the survey requires a survey link for the lab with attached token. In MOOC providers like Coursera, feedback is given with a comment or review and a rating. A registration is required, and the student needs to be enrolled in the course to give feedback. An advantage of the implementation used by the exchange platform is that the lab comment that requires authentication is not necessary, since the survey does not require authentication. Tokens can be generated as needed to be distributed to students. Version of the lab is also attached to the token so that analysis can be done separately on different version of the labs.

Analyzed and processed feedback in exchange platform can be seen publicly without authentication. The analyzed feedback is visualized and descriptive statistics is shown. In MOOC providers like Coursera, the latest comments on the course is shown in the course page, and its aggregated rating. The ratings are also shown in the course list. An advantage of the exchange platform is that visualization is available to provide more understanding of the survey results, where MOOC providers do not share the result of their survey. However, the current implementation of the exchange platform does not show the aggregated rating or result of the feedback in the catalog or in the lab page, requiring additional navigation to access the feedback results.

Browsing labs in the exchange platform is done in the catalog page, where it shows the list of available labs. Search function via keyword search and tag filter are provided. Similarly in MOOC platforms, keyword search and tag filter are also provided. Additionally, often the courses are further grouped, such as specializations, universities, or similar courses. The exchange platform does not offer this, and further groupings, such as multiple labs into a course should be considered.

Now we compare the sharing aspect of the exchange platform with Moodle and IEEE ComSoc Lab Exchange, due to a similar concept of lab sharing.

In order to share a lab in exchange platform, the lab is created from the *labsystem*, and exported to GitLab using an export script. In Moodle this is done by using Moodle's export tool, while in IEEE ComSoc Lab Exchange it is done by uploading the lab files directly. In its current implementation, the exchange platform has a disadvantage where the script has to be executed manually, and not integrated directly to the *labsystem*'s export tool. Moodle also has an advantage of selectively choosing which component of the course to be exported.

In the exchange platform, Moodle, and IEEE ComSoc Lab Exchange, a portal website is available to browse shared courses or labs. In all three, the labs needs to be downloaded and imported to their respective platform separately. It is not implemented in the exchange platform, but for future *labsystem* integration, importing labs can be done directly without browsing the portal, which make the process faster. Additionally, with Git, different versions of the lab can be checked out. However, similar with exporting labs, importing lab in exchange platform in the current implementation also requires a separate script to be executed, which adds additional process.

Versioning is one of exchange platform's advantages, which Moodle and IEEE ComSoc Lab Exchange do not provide. Versioning labs offers many possibilities: associating feedback to versions, comparing different versions of labs, checking out older versions, and forking labs to create a new lab. Since GitLab is used, features offered by GitLab can be used as well, such as issue tracking, branching, etc.

6.2 User Evaluation

An evaluation test is conducted to evaluate user experience in the exchange platform workflow. Two separate evaluation is conducted:

- Survey evaluation. Evaluate the survey workflow, starting from receiving survey url, submitting survey, and viewing survey result.
- *Labsystem* evaluation. Evaluate *labsystem* integration using the provided integration scripts.

Participants are given survey urls to multiple lab courses, from which they can submit survey for those courses. Participants are then asked to submit any data to the survey module. They are then instructed to view the survey result and analysis page to view the analyzed survey data. Additionally, participants are instructed to browse around the website and post comments in the lab pages. Finally, participants are asked to fill a user experience form to give their feedback for the whole process.

The evaluation form asks 8 questions; all questions other than the last are answered in a likert scale from *completely disagree* to *completely agree*, question 7 asks a rating, while the last question is a comment section. The questions are the following:

- The survey process is easy to follow and intuitive
- The user interface of the survey process is intuitive and informative
- My opinions on the subject is represented in the survey questions
- The data visualization is intuitive
- The data visualization helps me understand the data
- The data visualization shows information relevant to my interests
- Rate your experience
- Do you have a particular feedback?

A total of 15 participants filled in the user evaluation feedback form. The result is listed in Appendix B. The majority of participant express favorable user experience using the portal website. However, some remarked about the visualized graphs and how they are difficult to understand. These feedbacks are noted and considered as future enhancements for the portal website.

The other evaluation concerns with the integration with the *labsystem*. This is not made available for the participants of the evaluation since this requires the labs generated by the *labsystem*.

A supervisor is instructed to run the provided scripts and evaluate the *labsystem*. A lab is generated using the *labsystem*'s export tool. Using the export script, it is uploaded to the GitLab repository. Then the lab is modified, and using the update script, the changes are committed to GitLab. Finally, the lab is imported from GitLab to the *labsystem* using the import script. The integration scripts run successfully, and labs can be exported, updated, and imported using the provided scripts.

Additional evaluation by the next semester of iLab students are considered, but is outside the time scope of this thesis. The same instructions can be given to those students if this is to be conducted.

Chapter 7

Conclusion

In this thesis, a lab exchange platform is constructed by extending the functionalities of the *labsystem*. A GitLab repository is used to store labs. The *labsystem* exports and imports labs to a GitLab repository. A portal website is created to access labs in the GitLab, and to submit feedback for the labs. Result of the feedback is analyzed and available in the portal.

GitLab, while mostly known for software source code versioning, is shown to be capable of storing labs, solutions, and lab comments. Separate GitLab projects are used to store labs and their solutions. Forbidding access to these solutions is achieved by associating the projects in groups, and providing member access to said groups.

The implemented exchange platform fulfills goals to share labs to a community portal. Labs created by the *labsystem* are successfully exported to and imported from the GitLab repository. Using GitLab API, the portal shows labs stored in the repository. Provided integration scripts are tested successfully and integration with the running *labsystem* can be considered.

The platform also achieves feedback gathering, analysis, and visualization. Survey urls are distributed to students as primary method of gathering feedback for labs. Survey result is analyzed, with descriptive statistics and visualization shown to help understanding the data. A small initial test shows favorable opinion on the feedback delivering process. Visualization for the survey results receives criticism and should be improved.

Section 2.5 listed the questions this thesis aims to answer. A workflow consisting of the *labsystem* , GitLab, and portal website is made as a proof of concept for a lab exchange platform. Section 7.1 described on expanding the platform to multiple universities or institution. To ensure quality of the labs, feedback from students are gathered using survey and comments.

7.1 Future Work

Throughout the development of this thesis, several alternatives and features were considered, but did not make it into the final result. Here are some points to consider to develop the platform further.

- Directly integrate the *labsystem* scripts into the *labsystem*. In the current implementation, the scripts to push new, update existing labs, and import labs from the *labsystem* are run separately. This can be directly integrated to *labsystem*'s export functionality. Direct integration with the *labsystem* would make exporting and importing labs more convenient.
- Add emoji reactions to comments. Lab comments are implemented using GitLab comments, and emojis can be added (in GitLab terms, awarded) to each comments. Each emojis awarded are counted and shown beside their emoji icons. A similar implementation can be made in the lab comments. This is an additional feature to rate a comment, where later a high rated comment could be shown more prominently.
- Periodically remove unused or expired tokens from database. In the current implementation, expired tokens have to be removed manually: via the frontend portal, or using *mysql* queries. A better way would be to add a job on the web server to periodically run a task to clean these expired tokens.
- Word/tag cloud visualization for analyzing comments. In Chapter 4, we mentioned tag cloud to analyze comments but decided not to implement it. Regardless, this feature can be considered when adding more functionality on the survey result visualization.
- Integrate learning analytics from the *labsystem*. The *labsystem* performs some learning analytics as the students perform lab exercises. The results can be shown in the lab page in the portal website.
- Test implementation and deployment with multiple institutions. In the current implementation, there is only one institution or university (TUM) in the system. However, this proof of concept should work with multiple institution using the system. An approach to this problem is to use Git branches to store different versions of the lab for each institutions.
- Forking lab projects from GitLab. GitLab's fork project feature can be used to quickly create a new lab from another lab project. This can be useful if one wish to create a lab based on another.

Appendix A

Source Code

The source code of the implementation can be found in `https://gitlab.com/JulianElda/Exchange_Platform_Frontend`. The instructions can be found in the wiki page in `https://gitlab.com/JulianElda/Exchange_Platform_Frontend/wikis/home`. More detailed comments and instructions of the implementation can be found in the source code.

Additionally, the thesis project can be found in `https://gitlab.dev.ds2os.org/thesis/ma_polar`. A mirror of the source code is also available in `https://gitlab.dev.ds2os.org/polar/exchange_platform_frontend`.

Appendix B

Survey Evaluation

The survey evaluation's instruction is available in
`https://gitlab.com/JulianElda/Exchange_Platform_Frontend/wikis/evaluation`

The instructions for *labsystem* evaluation is available in `https://gitlab.com/JulianElda/Exchange_Platform_Frontend/wikis/labsystem`

The user experience form is available in
`https://docs.google.com/forms/d/1vwQO1hfVE8_kzXxsr0vbytrH5MIG1w7xV6hgjg4qQVw`

The result of the user experience survey is available in
`https://docs.google.com/forms/d/1vwQO1hfVE8_kzXxsr0vbytrH5MIG1w7xV6hgjg4qQVw/viewanalytics`

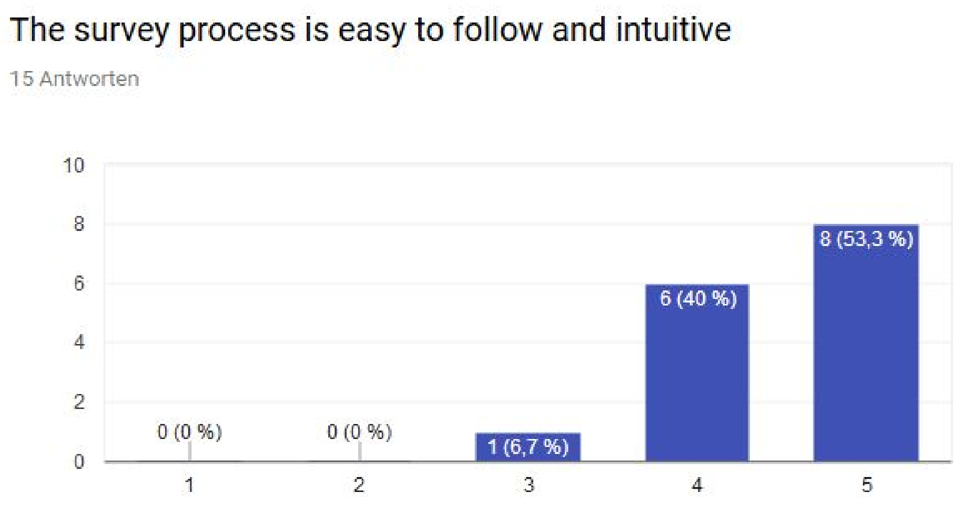


Figure B.1: Evaluation survey result, part 1

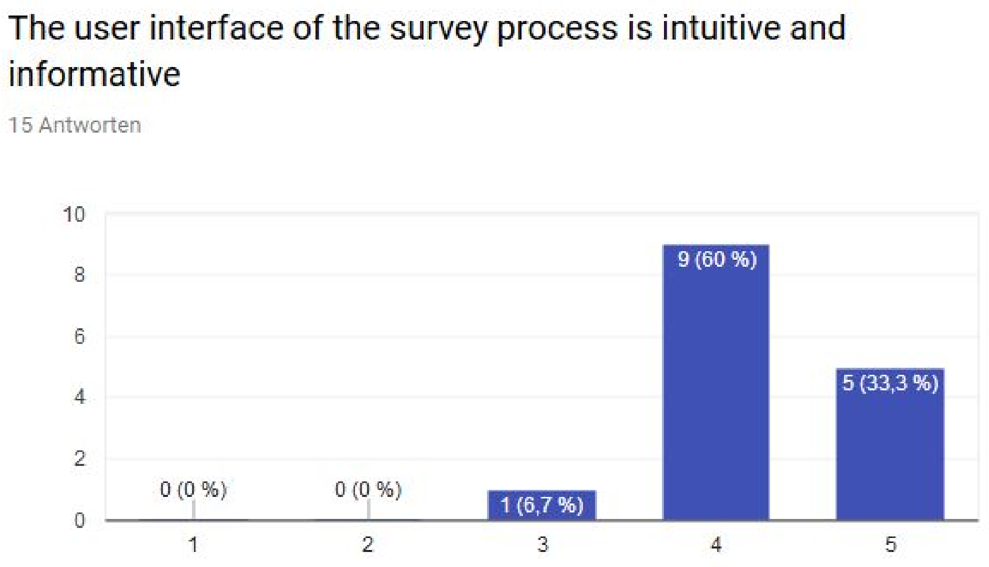


Figure B.2: Evaluation survey result, part 2

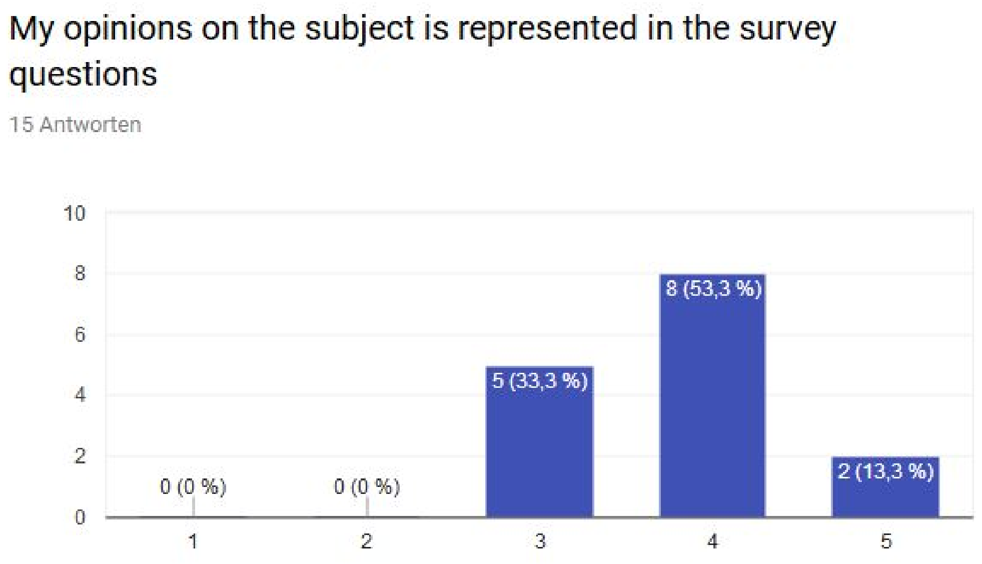


Figure B.3: Evaluation survey result, part 3

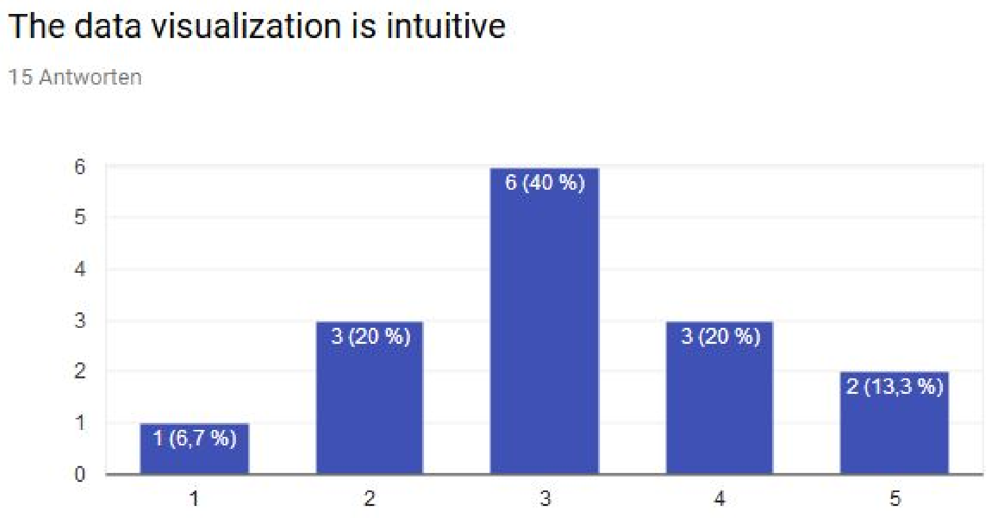


Figure B.4: Evaluation survey result, part 4

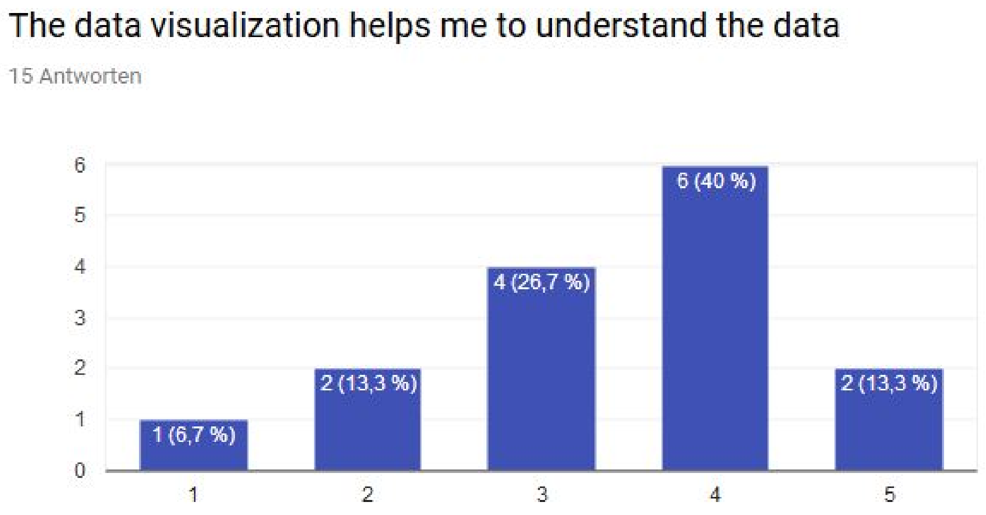


Figure B.5: Evaluation survey result, part 5

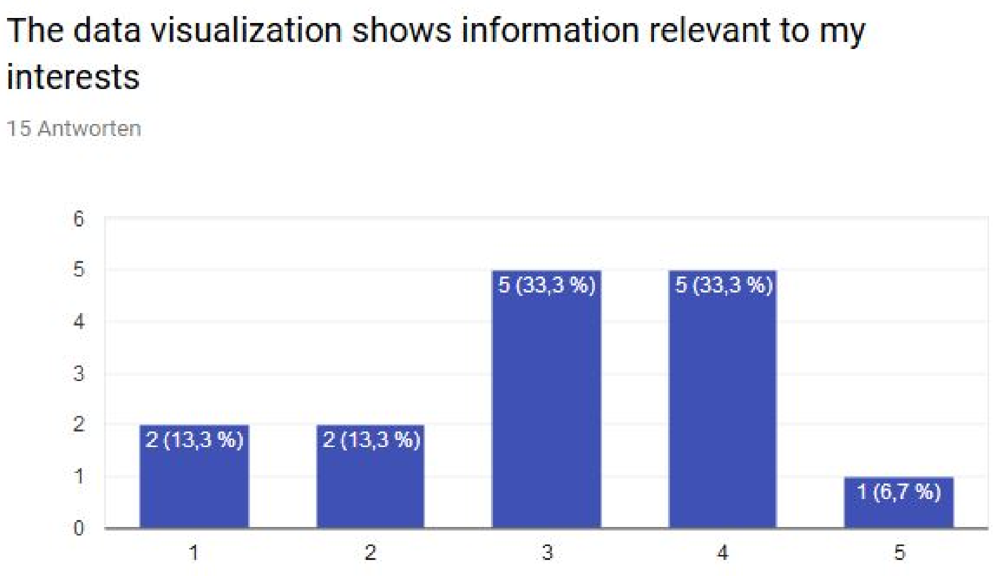


Figure B.6: Evaluation survey result, part 6

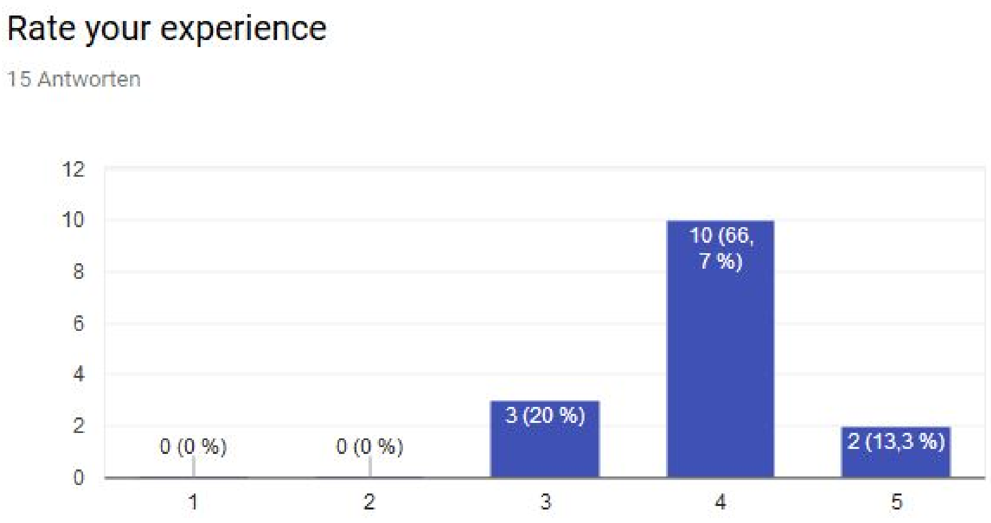


Figure B.7: Evaluation survey result, part 7

Bibliography

[1] A. Hargreaves, *Teaching in the knowledge society: Education in the age of insecurity.* Teachers College Press, 2003.

[2] M. Alkhattabi, D. Neagu, and A. Cullen, "Assessing information quality of e-learning systems: a web mining approach," *Computers in Human Behavior*, vol. 27, no. 2, pp. 862–873, 2011.

[3] M. Castells, *The Internet galaxy: Reflections on the Internet, business, and society.* Oxford University Press on Demand, 2002.

[4] S. Downes, "E-learning 2.0.," *Elearn magazine*, vol. 2005, no. 10, p. 1, 2005.

[5] Y. E. Kalay, "Virtual learning environments," *Journal of Information Technology in Construction (ITcon)*, vol. 9, no. 13, pp. 195–207, 2004.

[6] R. O'Leary and A. Ramsden, "Virtual learning environments," *Learning and Teaching Support Network Generic Centre/ALT Guides, LTSN. Retrieved July*, vol. 12, p. 2005, 2002.

[7] R. C. Clark and R. E. Mayer, *E-learning and the science of instruction: Proven guidelines for consumers and designers of multimedia learning.* John Wiley & Sons, 2016.

[8] Moodle, "Moodle.net," 2017. [Online, last accessed 12.09.2017; `https://moodle.net`].

[9] J. Donovan, C. E. Mader, and J. Shinsky, "Constructive student feedback: Online vs. traditional course evaluations.," *Journal of Interactive Online Learning*, vol. 5, no. 3, pp. 283–296, 2006.

[10] S. E. Sampson, "Gathering customer feedback via the internet: instruments and prospects," *Industrial Management & Data Systems*, vol. 98, no. 2, pp. 71–82, 1998.

[11] K. B. Wright, "Researching internet-based populations: Advantages and disadvantages of online survey research, online questionnaire authoring software packages, and web survey services," *Journal of Computer-Mediated Communication*, vol. 10, no. 3, pp. 00–00, 2005.

[12] W. C. Schmidt, “World-wide web survey research: Benefits, potential problems, and solutions,” *Behavior Research Methods*, vol. 29, no. 2, pp. 274–279, 1997.

[13] M. D. Kaplowitz, T. D. Hadlock, and R. Levine, “A comparison of web and mail survey response rates,” *Public opinion quarterly*, vol. 68, no. 1, pp. 94–101, 2004.

[14] N. Kumar and I. Benbasat, “Shopping as experience and website as a social actor: web interface design and para-social presence,” *ICIS 2001 Proceedings*, p. 54, 2001.

[15] R. Cheng and J. Vassileva, “Design and evaluation of an adaptive incentive mechanism for sustained educational online communities,” *User Modeling and User-Adapted Interaction*, vol. 16, no. 3-4, pp. 321–348, 2006.

[16] P. A. Tess, “The role of social media in higher education classes (real and virtual)–a literature review,” *Computers in Human Behavior*, vol. 29, no. 5, pp. A60–A68, 2013.

[17] G. Siemens and P. Long, “Penetrating the fog: Analytics in learning and education.,” *EDUCAUSE review*, vol. 46, no. 5, p. 30, 2011.

[18] E. Duval, “Attention please!: learning analytics for visualization and recommendation,” in *Proceedings of the 1st International Conference on Learning Analytics and Knowledge*, pp. 9–17, ACM, 2011.

[19] B. Burdeva, “Improving elearning through statistical feedback,” 2017.

[20] K. Ram, “Git can facilitate greater reproducibility and increased transparency in science,” *Source code for biology and medicine*, vol. 8, no. 1, p. 7, 2013.

[21] K. Brandl, “Are you ready to “moodle”,” *Language Learning & Technology*, vol. 9, no. 2, pp. 16–23, 2005.

[22] Moodle, “Moodle.net: Course approval criteria,” 2017. [Online, last accessed 12.09.2017; `https://moodle.net/mod/page/view.php?id=2`].

[23] A. Kaushik, *Web analytics: An hour a day (W/Cd).* John Wiley & Sons, 2007.

[24] S. A. Miller, *Piwik Web Analytics Essentials.* Packt Publishing Ltd, 2012.

[25] S. Chacon and B. Straub, *Pro git.* Apress, 2014.

[26] J. Loeliger, “Collaborating with git,” *Linux Magazine, June*, 2006.

[27] J. M. Hethey, *GitLab Repository Management.* Packt Publishing Ltd, 2013.

[28] GitHub, “Github api,” 2017. [Online, last accessed 12.09.2017; `https://developer.github.com/v3/`].

[29] GitLab, “Gitlab api,” 2017. [Online, last accessed 12.09.2017; `https://docs.gitlab.com/ee/api/`].

[30] R. Rivard, “Measuring the mooc dropout rate,” *Inside Higher Ed*, vol. 8, p. 2013, 2013.

[31] T. Daradoumis, R. Bassi, F. Xhafa, and S. Caballé, “A review on massive e-learning (mooc) design, delivery and assessment,” in *P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), 2013 Eighth International Conference on*, pp. 208–213, IEEE, 2013.

[32] S. Taneja and A. Goel, “Mooc providers and their strategies,” *International Journal of Computer Science and Mobile Computing*, vol. 3, no. 5, pp. 222–228, 2014.

[33] Coursera, “Privacy policy,” 2016. [Online, last accessed 12.09.2017; `https://www.coursera.org/about/privacy`].

[34] edx, “edx privacy policy,” 2016. [Online, last accessed 12.09.2017; `https://www.edx.org/edx-privacy-policy`].

[35] Yale, “Yalecourses,” 2017. [Online, last accessed 12.09.2017; `https://www.youtube.com/channel/UC4EY_qnSeAP1xGsh61eOoJA`].

[36] Google, “Privacy policy,” 2016. [Online, last accessed 12.09.2017; `https://www.google.com/intl/en/policies/privacy/`].

[37] F. L. Cerdà and N. C. Planas, “Facebook’s potential for collaborative e-learning,” *Revista de Universidad y Sociedad del Conocimiento*, vol. 8, no. 2, pp. 197–210, 2011.

[38] Facebook, “Data policy,” 2017. [Online, last accessed 12.09.2017; `https://www.facebook.com/policy.php`].

[39] Facebook, “Page roles,” 2017. [Online, last accessed 12.09.2017; `https://www.facebook.com/help/1206330326045914/?helpref=hc_fnav`].

[40] Moodle, “Question permissions explained with diagrams - moodledocs,” 2017. [Online, last accessed 12.09.2017; `https://docs.moodle.org/24/en/Question_permissions_explained_with_diagrams`].

[41] Moodle, “Standard roles - moodledocs,” 2017. [Online, last accessed 12.09.2017; `https://docs.moodle.org/32/en/Standard_roles`].

[42] Moodle, “Permissions - moodledocs,” 2017. [Online, last accessed 12.09.2017; `https://docs.moodle.org/32/en/Permissions`].

[43] Moodle, “Global search - moodledocs,” 2017. [Online, last accessed 12.09.2017; `https://docs.moodle.org/31/en/Global_search`].

[44] Moodle, “Site-wide reports - moodledocs,” 2017. [Online, last accessed 12.09.2017; `https://docs.moodle.org/32/en/Site-wide_reports`].

[45] Moodle, "Moodle plugins directory: overview statistics," 2017. [Online, last accessed 12.09.2017; `https://moodle.org/plugins/report_overviewstats`].

[46] R. Bowley, E. Luther, and D. G. Michelson, "The comsoc hands-on lab exchange," *IEEE Communications Magazine*, 2016.

[47] IEEE, "Hands on lab exchange," 2017. [Online, last accessed 12.09.2017, last accessed 12.09.2017; `http://labs.comsoc.org/`].

[48] O. E. Resources, "Oer commons," 2017. [Online, last accessed 12.09.2017; `https://www.oercommons.org/about`].

[49] O. E. Resources, "Oer commons," 2017. [Online, last accessed 12.09.2017; `https://www.oercommons.org/oer`].

[50] GitHub, "Working with large files," 2017. [Online, last accessed 12.09.2017; `https://help.github.com/articles/working-with-large-files/`].

[51] GitLab, "Announcing git lfs support in gitlab," 2015. [Online, last accessed 12.09.2017; `https://about.gitlab.com/2015/11/23/announcing-git-lfs-support-in-gitlab/`].

[52] J. T. Richardson, "Instruments for obtaining student feedback: A review of the literature," *Assessment & Evaluation in Higher Education*, vol. 30, no. 4, pp. 387–415, 2005.

[53] N. Korfiatis, E. GarcíA-Bariocanal, and S. Sánchez-Alonso, "Evaluating content quality and helpfulness of online product reviews: The interplay of review helpfulness vs. review content," *Electronic Commerce Research and Applications*, vol. 11, no. 3, pp. 205–217, 2012.

[54] "Survey analysis guidelines," *American Association of Community College*, vol. 6.

[55] N. B. Robbins, R. M. Heiberger, *et al.*, "Plotting likert and other rating scales," in *Proceedings of the 2011 Joint Statistical Meeting*, pp. 1058–1066, 2011.

[56] R. M. Heiberger, N. B. Robbins, *et al.*, "Design of diverging stacked bar charts for likert scales and other applications," *Journal of Statistical Software*, vol. 57, no. 5, pp. 1–32, 2014.

[57] S. Lohmann, J. Ziegler, and L. Tetzlaff, "Comparison of tag cloud layouts: Task-related performance and visual exploration," in *IFIP Conference on Human-Computer Interaction*, pp. 392–404, Springer, 2009.

[58] W. Cui, Y. Wu, S. Liu, F. Wei, M. X. Zhou, and H. Qu, "Context preserving dynamic word cloud visualization," in *Visualization Symposium (PacificVis), 2010 IEEE Pacific*, pp. 121–128, IEEE, 2010.

[59] L. Dabbish, C. Stuart, J. Tsay, and J. Herbsleb, "Social coding in github: transparency and collaboration in an open software repository," in *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work*, pp. 1277–1286, ACM, 2012.

[60] G. Gousios, B. Vasilescu, A. Serebrenik, and A. Zaidman, "Lean ghtorrent: Github data on demand," in *Proceedings of the 11th working conference on mining software repositories*, pp. 384–387, ACM, 2014.

[61] D. Balla, "State of app development in 2016," 2017. [Online, last accessed 12.09.2017; `http://blog.bitrise.io/2017/01/27/state-of-app-development-in-2016.html`].

[62] GitLab, "What's next for gitlab ci," 2017. [Online, last accessed 12.09.2017; `https://about.gitlab.com/2017/06/29/whats-next-for-gitlab-ci/`].

[63] D. Kohn, "The 30 highest velocity open source projects," 2017. [Online, last accessed 12.09.2017; `https://www.cncf.io/blog/2017/06/05/30-highest-velocity-open-source-projects/`].

[64] GitLab, "Gitlab eep vs. github enterprise," 2017. [Online, last accessed 12.09.2017; `https://about.gitlab.com/comparison/gitlab-ee-vs-github-enterprise.html`].

[65] GitLab, "Gitlab user permissions," 2017. [Online, last accessed 12.09.2017; `https://docs.gitlab.com/ee/user/permissions`].

[66] GitLab, "Gitlab groups," 2017. [Online, last accessed 12.09.2017; `https://docs.gitlab.com/ce/workflow/groups.html`].

[67] GitLab, "Gitlab issues," 2017. [Online, last accessed 12.09.2017; `https://docs.gitlab.com/ee/user/project/issues/index.html`].

[68] GitLab, "Gitlab installation," 2017. [Online, last accessed 12.09.2017; `https://about.gitlab.com/installation`].

[69] GitLab, "Gitlab configuration options," 2017. [Online, last accessed 12.09.2017; `https://docs.gitlab.com/omnibus/settings/configuration.html`].