

communicated by:
Lehr- und Forschungsgebiet Informatik 9
Prof Dr. Ulrik Schroeder



RWTHAACHEN
UNIVERSITY

The present work was submitted to Learning Technologies Research Group
Diese Arbeit wurde vorgelegt am Lehr- und Forschungsgebiet Informatik 9

Development of an Exchange Platform for WebWriter-Explorables
Entwicklung einer Austauschplattform für WebWriter-Explorables

Master-Thesis
Masterarbeit

presented by / von
Troll, Marc
394593

Prof. Dr.-Ing. Ulrik Schroeder
Prof. Dr. Jan Borchers

supervised by
Frederic Salmen

Aachen, November 11, 2025

Contents

List of Figures	iii
List of Tables	iv
List of Listings	v
1. Introduction	2
1.1. Goal of This Thesis	2
1.2. Structure of This Thesis	3
1.3. Fundamental Concepts and Definitions	4
1.4. Related Work	5
2. Concept	8
2.1. Requirements Analysis	8
2.1.1. Stakeholder Definition	9
2.1.2. Functional Requirements	9
2.1.3. Non-Functional Requirements	11
2.2. Usage in Learning Environments	12
2.3. Prototype Design	13
2.3.1. Inspired by existing software	14
3. Implementation	15
3.1. System Architecture	15
3.2. Technology Stack	16
3.2.1. Web Frameworks	16
3.2.2. Server-Side Rendering	17
3.2.3. Nuxt	18
3.2.4. PocketBase	18
3.3. Data Model and Integration	19
3.4. Implementation Details	21
3.4.1. Project and Folder Structure	22
3.4.2. External Libraries and Components	23
3.4.3. Integration of PocketBase in the Application	24
3.4.4. Role-Based Access in Groups	26
3.5. Final Result	28
3.6. Design Decisions, Challenges and Improvements	29
4. Evaluation	32
4.1. Theoretical Foundations	32
4.2. Preparation	33

4.3. Pre-Study	36
4.4. Main Study	37
4.5. Results of the Main Study	39
4.6. Interpretation and Discussion	41
5. Conclusion	43
5.1. Future Work	43
Appendix	44
A. Bibliography	45
B. Online References	47
C. Listings	51
D. Digital Appendix	54
E. Assistive Tools	55

List of Figures

1.1. Overview on the WebWriter ecosystem. Visualization from Frederic Salmen. The part marked in green currently does not exist and will be implemented within this thesis.	3
2.1. Figma prototypes for the software project. The difference between these prototypes is the navigation: (a) uses a banner, while (b) uses two menu links to switch between exchange and learning platform.	13
3.1. Architecture of the software project. Communication between API-Service and external API is implemented using HTTP/REST.	15
3.2. Database Model/Schema implemented in PocketBase	20
3.3. Example visualization of a hierarchical group structure including possible group assignments for a single user	28
3.4. Reduced collection of screenshots that show different types of pages. . . .	30
4.1. Members List of a Learning Group	38
4.2. Scale Means and Importance Ratings of the selected UEQ+ categories for the prototype evaluation	40

List of Tables

3.1. List of permissions granted by role in a specific group	27
--	----

List of Listings

3.1. Example implementation of a page using the previously implemented composables. The example is a short version of the main page.	28
C.1. Implementation of the <code>CookieAuthStore</code>	51
C.2. Implementation of the <code>Composable</code>	53

Abstract

In educational contexts, worksheets are used for teaching and learning. Such worksheets are created and provided on platforms and marketplaces to third persons. With WebWriter exists a web-based authoring tool for creating digital and interactive teaching materials - so-called Explorables. Currently, WebWriter has no platform for sharing Explorables to other educators and learners.

Goal of this thesis is to create a concept and implement a prototype for such a platform. The platform combines community-based with course-based functionalities and integrates them into the ecosystem of WebWriter. Community-based functionalities include publishing, searching, and rating Explorables while course-based features provide a learning management system with learning groups. The development of the platform includes to first collect requirements, before implementing the platform. For the implementation, technologies have to be selected carefully.

At the end of this thesis, the prototype is evaluated using a user study. The study concentrates on the user experience of educators. For the evaluation, the User Experience Questionnaire Plus is used. The results are presented and an outlook for future development is provided.

Chapter 1 Introduction

Teachers often use worksheets for education in schools or universities. These worksheets can be edited on paper as well as on digital devices such as tablets or notebooks. Especially since the Coronavirus pandemic, the use of digital technologies has increased [JKT21]. In contrast to paper worksheets with static content, the authoring tool WebWriter provides possibilities for creating digital and interactive worksheets. Worksheets created with WebWriter - called Explorables - are exported as HTML files, which can be shared with and edited by other persons such as teachers or students. Editing does not require any additional software but a standard web browser.

Currently, Explorables created with WebWriter can be stored locally on the device of the user or in the WebWriter Cloud. However, both storage options do not allow third persons to search for and use Explorables. With this master's thesis a prototype for such a platform is supposed to be implemented. The software application accesses the database of the WebWriter Cloud and therefore allows searching and using Explorables that are stored inside that database using a web application.

This first chapter provides an introduction into the topic, especially WebWriter, classifies and describes the goal of my work and gives an overview on the structure of this thesis.

1.1. Goal of This Thesis

The current system architecture of WebWriter consists of different components. Figure 1.1 shows a visualization, which includes existing components of the WebWriter system architecture, but also - marked in green - a platform. This is, where the goal of my thesis comes in.

Goal of this thesis is the development and evaluation of a prototype for an exchange platform for digital learning resources. Primarily, the exchange platform is supposed to be used with WebWriter. It therefore integrates within WebWriter's existing ecosystem and reuses technologies and systems such as databases, that are already existing within the scope of WebWriter.

The software application consists of two parts:

1. **Exchange Platform:** This part is similar to a marketplace. The exchange platform is supposed to make Explorables accessible to other users. Beside searching for Explorables, the exchange platform shall contain community-based features: Rating and reviewing Explorables, following other users, as well as saving and organizing Explorables for later on.

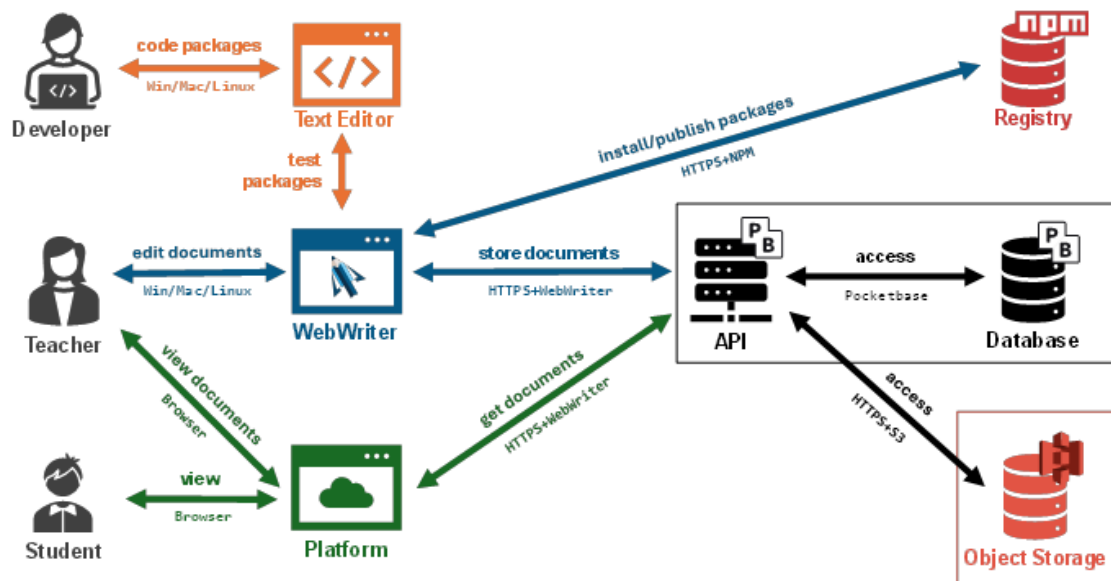


Figure 1.1.: Overview on the WebWriter ecosystem. Visualization from Frederic Salmen. The part marked in green currently does not exist and will be implemented within this thesis.

2. **Learning Platform:** Within the learning platform, users (especially educators) shall organize classes and their students and provide them with learning materials. For this prototype, this is limited to Explorables, that are stored inside the WebWriter Cloud.

Both platforms are combined within a single web application, which can be deployed on any server. Access to the application is granted every user with a modern web browser, although users may need to create an account before accessing some features.

The work should contribute to the research question of "How can the remix workflow be supported by an exchange platform for Explorables?". Thus, the research of this thesis is particularly in the field of web and learning technologies, eLearning as well as human-computer interaction.

1.2. Structure of This Thesis

The thesis consists of five chapters: Introduction, concept, implementation, evaluation, and conclusion. The first one gives an introduction to the topic, provides fundamental concepts and definitions and puts the thesis into context of what others have already done in this field.

In the next chapter - concept - possible users are defined, requirements are collected and a first design prototype is discussed. Based on the concept, the actual prototype will be implemented in chapter 3. The used technologies are discussed, adjustments to the existing infrastructure are provided, and problems and challenges during the implementation are presented.

The fourth chapter includes the evaluation of the implemented prototype of the software application. For this, the standardized UEQ+ (User Experience Questionnaire Plus) will be used. The results are discussed afterwards. In the last chapter, a final conclusion is drawn. In addition, the work and its results are summarized and an outlook on future developments is given.

1.3. Fundamental Concepts and Definitions

In this thesis, some terms and definitions are used. In the following, the most important terms are defined and described. The goal is to provide a common knowledge on the used terminology for the following chapters.

Library

A library is a collection of code. It provides default implementations for various tasks depending on the purpose of the library. A developer can use the implementation of a library in its code. By using a library, the amount of time being required for the implementation as well as the error rate are reduced. A library can not run on its own but must be used within and called by another software. [49]

(Web) Framework

A framework or web framework is a special form of a library. It can run alone and "provides a structure for the project. Using frameworks, the developer must not start from the beginning, but he can use the existing structure to implement his own code based on the framework." [Tro23] The same applies to frameworks as to libraries: The framework "provides well-tested default implementations to the developer, which the developer can use to save time and reduce the risk of errors." [Tro23]

Single-Page Application

"Single Page Applications (SPA) are applications that are loaded only once. After the page has been loaded, the user can navigate inside the application which updates the content inside the browser of the user. If required, additional data (e.g. data stored inside databases) can be loaded from a server by requesting them using corresponding methods from inside the SPA." [Tro23]

REST API

REST stands for REpresentational State Transfer and defines six guiding principles for the development of web services. A web service fulfilling these principles is called REST API (Application Programming Interface) or RESTful API. RESTful APIs provide a uniform interface, that can be used by clients to identify and access data on a server. Furthermore, REST is stateless, meaning that all relevant information to complete a request must be included in the request. Responses by such APIs may allow (or prohibit) caching the results on the client, which may reduce the amount of requests sent to the server. Most times, HTTP is used as the protocol for implementing REST APIs, although the protocol is not defined in the guidelines and is therefore not limited to HTTP. [40]

Git / GitHub

Git is a version control system. It defines various operations that allow developers to track and manage multiple versions of their source code. The source code of a project is stored in so-called repositories, which keeps track on changes to the files associated with it. Git allows for easy collaboration between multiple developers by using branching, which creates copies of the source code that can be worked on independently. GitHub is a web-based platform, which is built upon Git. It therefore implements the functionalities of Git, stores the source code of various projects on its servers and makes it accessible to other developers. [2]

1.4. Related Work

This thesis is based in parts on existing software or may draw inspiration from them. It may also make use of (scientific) findings already collected by others. This section introduces the context of the thesis and presents software and research related to this work.

One of the most relevant software systems in the context of this thesis is WebWriter. WebWriter is a web-based authoring system for creating digital and interactive worksheets. It is publicly accessible and can be used through a standard web browser, without the need to install additional software.

Worksheets created with WebWriter are called Explorables. Explorables are "emphasizing the learner's active relationship with the resource" [SRS23]. Explorables are designed to be open, accessible, multimodal, interactive, web-based, and file-oriented. [SRS23] [16]

These characteristics contribute to the principle of open educational resources (OER), which are resources that "are licensed with copyright licenses that provide permission for everyone to participate in the 5R activities - retain, reuse, revise, remix, and redistribute" [WHI18]. From this the remix workflow is defined, which allows educators to use and change existing learning materials and create new ones, differentiated for their learners and their skills. [WHI18] [SRS23]

From a technical view, WebWriter is implemented using modern web technologies, where Web Components are the basis WebWriter builds on. Web Components [56] allow the encapsulation of functionality as custom HTML elements written in JavaScript. These custom elements are used to implement so-called widgets [57]. Widgets are interactive multimedia elements, which extend the basic feature set of WebWriter. With the concept of Web Components, widgets can be flexibly combined and reused in multiple HTML documents.

Currently, 19 widgets are published on the official WebWriter package list [58]. This includes widgets such as quizzes, interactive and executable code cells, visualizations for neural networks, interactive videos, and parts of the mathematics learning app GeoGebra. The list of widgets is growing. At the beginning of the thesis in April, this list contained no more than 16 items.

Using these widgets, users (especially educators) can author their own Explorables directly within WebWriter. Explorables in general are portable and standalone HTML documents. Being standalone means that all dependencies and assets are stored within a single file and no active internet connection is required [SRS23] [SRS24]. As a result, Explorables can be used online and offline, which also makes them applicable in teaching and in situations with limited connectivity.

In a previous thesis, Allius [All23] developed a server component for WebWriter. This server component was designed as a central server storage tool for Explorables. It was accessible through a RESTful API allowing uploading, updating, reading, and deleting Explorables. However, some functionalities were missing: "The solution does not provide detailed access restrictions, e.g. directly sharing the file with a set of users (student classes, other teachers, ...). Implementing so would require a full user management." [All23]

In the meanwhile, this server component is no longer used and replaced by a PocketBase [37] instance. PocketBase is a software that helps managing databases and allows accessing this data through REST. PocketBase includes but is not limited to data management, user management, and authentication. It is furthermore extendable using Go or JavaScript using so called hooks. More details about PocketBase and its usage in the context of this thesis are provided within the implementation chapter.

My thesis project consists of a combination of two types of platforms: exchange platform as well as learning platform. For both of these types, various platforms with their own advantages and disadvantages exist. In the following, I will present some of them with regard to their functionality and discuss, in which extent they may influence the implementation of the prototype within this thesis.

A first example that I will present is eduki [15]. Eduki is a web-based platform primarily targeting authors and educators to upload and share learning resources with each other. It therefore belongs to the category of exchange platforms. Users of eduki can search for, download and rate teaching materials, that was added by others. Learning materials are structured by and can be filtered for various characteristics such as the type of school, class, subject and other. Additionally, eduki enables authors of learning resources to charge money for their materials. Authors can thereby get paid for their work and effort that was put into the making of teaching materials.

However, eduki does only provide the exploration and finding of learning resources. To digitally organize classes and provide learning resources to students, a teacher requires a second platform: This can be chat-like apps but also learning platforms like Moodle.

Moodle [29] is an example for a so-called learning management system (LMS). It provides functionalities to organize courses and share learning resources within these courses. Some learning resources such as quizzes can also be directly integrated into Moodle with automatic correction of the solution provided by learners. Moodle can furthermore be extended and customized using plugins.

Moodle, for example, is also used in Northrhine-Westfalia (NRW), Germany, providing a LMS to schools - the digital learning platform LOGINEO NRW LMS [26]. LOGINEO

NRW LMS was introduced in June 2020, short time after the beginning of the Coronavirus pandemic. The Coronavirus pandemic boosted digital teaching and learning, where face-to-face teaching was digitally substituted. In February 2023 LOGINEO NRW LMS was used by 2600 of more than 5000 schools in NRW [41] [fOKF23] [GRB21]. However, in the case of LOGINEO NRW LMS some users state, that - in general - the LMS based on Moodle is okayish to be used but some tasks in managing schools, courses, and students are complex or time-consuming. [fOKF23]

Taking a look at the learning management system provided to schools in Schleswig-Holstein, the amount of students actually using the LMS decreased from 25% in 2022 to only 14% of all students in 2024. Exact reasons are not given [Hab24]. In an interview with Prof. Kerres, who is a professor of education and holds the chair of Media Didactics and Knowledge Management at the University of Duisburg-Essen in Germany, he suggests an explanation for these kind of observations: He explains that the Coronavirus pandemic forced people to use digital solutions. He suggests that now, since the pandemic is over and there is no longer any need to use digital solutions, many teachers are going back to their pre-pandemic routines and digitalization is being neglected again. [11]

Although the current usage of digital learning resources and learning management systems in schools seems not very promising, the new German federal government found a new federal ministry: The Federal Ministry for Digital and State Modernization [7]. The foundation of the new federal ministry is a clear sign for further enhancements in digital resources. In addition to the "Digitalpakt Schule" [12], which was a funding program in Germany granting 6.5 billion Euro for schools to invest into their digital equipment, it is clear that digitalization will continue to advance in Germany and will be an integral part of teaching in the long term.

Chapter 2 Concept

The first step, when creating a new software, is to think about who will actually use the software. It is mandatory to know about the target groups (stakeholders) in order to design an appropriate interface for them. After the stakeholders are defined, requirements are collected, which the software is supposed to fulfill. Based on the previously collected knowledge on target groups and requirements, some rapid prototypes can be created for a first discussion before starting the implementation.

This project consists of two types of platforms: exchange platform and learning platform. The exchange platform defines community-based functionalities and the learning platform defines course-based functionalities. Both parts of the software - the exchange platform as well as the learning platform - should be combined into one single, easily accessible web application.

2.1. Requirements Analysis

Requirements analysis is an essential part in software development. It includes the process before the implementation of a software, where user's needs and requirements are collected. In a first step, background information on possible users are collected. These users, also known as stakeholders, define different requirements based on their roles, responsibilities, skills, interests, and expectations. These requirements are collected for each stakeholder group and define all the functions of the final software project. [MB02] [SDW⁺10]

Requirements are categorized in functional and non-functional requirements. Functional requirements describe the functions the software must provide in order to perform the intended tasks. Non-functional requirements on the other hand define quality requirements such as usability, performance, reliability, or maintainability. [SDW⁺10] [iee18]

A basic set of requirements and stakeholders were given by the tender for this thesis. The basic set of requirements and stakeholders was used to create a more detailed list. First, stakeholders were defined. Afterwards, functional and non-functional requirements were specified. Given all the stakeholders and requirements, a first prototype was developed. However, that prototype was not really detailed for a simple reason: The exchange platform uses components and implements features that are already known from software applications (e.g. Amazon [4] or YouTube [60]), that are used by millions or billions of users. [30] [5]

2.1.1. Stakeholder Definition

At the beginning five stakeholders were discovered: Students, teachers, authors, administrators, and software engineers. In a later discussion, it turned out that a differentiation between authors and teachers is not necessary. Every teacher has the opportunity to upload and publish Explorables they have created themselves and thus become an author. Therefore, distinguishing between teachers and authors brings little added value on the one hand and on the other hand increases the effort in development and maintenance and could cause confusion for users. In the end, teachers and students are supposed to be everyday users while administrators and software engineers might not use the platform so often. [Cla95]

Teachers are persons that are supposed to use the exchange platform as well as the learning platform. Their workflow includes especially searching and remixing Explorables in the exchange platform, managing learner groups (in the following they are simply called groups) inside the learning platform, and sharing Explorables within these groups. Managing learner groups means especially organizing learners and students inside these groups.

Software engineers are required for further improvements, bug fixes and maintenance of this software project later on. A simple, well-organized software architecture as well as clear, documented, and reusable code helps software engineers to familiarize with and improve the existing project. Although, software engineers do not define explicit functionalities for the software project, they must be recognized as a stakeholder.

Administrators are primarily persons of schools or organizations that are responsible for managing groups of learners. They should be provided with functionalities and interfaces which ease and possibly allow automation of the management of students in any kind of organization.

Students are these users that should have access primarily to groups they belong to. Within these groups they should have possibilities to access and download Explorables that are shared with the group. Therefore, students are mainly using the learning platform of the software project, although access to the exchange platform is not prohibited. Then, students can also search for, download, and solve Explorables on their own.

2.1.2. Functional Requirements

The following list provides an overview on the features being implemented. It also states, which stakeholders are supposed to use the corresponding feature. The list was created using the basic set of requirements and was extended throughout the implementation of the software project. The functional requirements can also be categorized into different categories: Explorables, profiles, and groups. Explorables and profiles can be categorized as features for the exchange platform. Groups, on the other hand, can be categorized as features for the learning platform.

The list provides information on those stakeholders, that are primarily meant to use the corresponding feature. But, the feature is not limited to be used by only the named stakeholders. The software system itself will not distinguish between stakeholders and provide explicit barriers, such that a stakeholder can only access those features that are defined for it. All users can access all features, regardless of which stakeholder group they belong to.

Starting with the exchange platform, the user must be able to search for and find Explorables. Doing so, requires pages for Explorable lists (overview pages), text search, as well as viewing Explorable details (detail pages). All these pages are meant to be used especially by teachers fulfilling the task of gathering learning materials and connecting with each other.

Using the overview pages and the text search users get a list of Explorables. While the overview pages show recommended Explorables, the text search page must provide Explorables that fit to the user's search term. When listing Explorables on these pages, compact cards with the most important information such as title, author, overall rating, and a short description should be shown. The overall rating is the average rating over all ratings for that Explorable.

On the detail pages users should be able to find all details regarding a specific Explorable. This includes at least the title, author, overall rating, the full description, meta information, and reviews. Reviews include the review itself, the rating, and author information. If a user opens the detail page of one of its own Explorables, the user must be able to edit the Explorable including title, description, and other meta information. These changes must also be written to the corresponding Explorable file. Users must also have the possibility to create and edit their own reviews and ratings for an Explorable on the detail pages. Users are not allowed to rate their own Explorables. Reviews will not be automatically checked for sufficient quality or if a reviewer actually used the corresponding Explorable. In reviews, a rating is required but not a comment. A comment is optional. The rating of an Explorable is on a scale from 1 to 5. A user can only rate each Explorable once.

In addition, users should be able to view other user's profiles. A profile is the publicly accessible information of a user account. The profile page should contain possibilities to follow and unfollow other users, see a list of who is following the user and whom is the user following, as well as a list of Explorables the user has created. Using the follow system, users can keep updated on new Explorables from users they are following. A user can not follow himself.

Next, taking a look at the functional requirements for the learning application: In the learning application users should be able to create groups. Groups are the basic system to organize learning groups and students. Groups can contain content or other groups (so called subgroups). Groups are therefore organized hierarchically.

In a group there must be three roles for access management: Administrators (or short admins), managers, and members. The roles define access rights for the corresponding group. I will later discuss details on the role assignment. Starting with the lowest role, members should be able to access a group and see content that is published to the group.

For this software project, members should not have any other access rights. Managers are able to access (view) and modify (edit) a group (e.g. change the name of the group), create subgroups within a group and manage content including creating, editing, deleting, and publishing content. Admins should be able to delete, edit, and view a group. Both, managers and admins can add users to the group. The default role of new group members is the "members" role. Managers can then assign the "members" and "managers" roles to users, while admins can assign the "members", "managers", and "admins" role to users.

Content in groups can be managed by admins and managers. Both roles are granted the same access rights with regard to content management. Possible content types include but are not limited to headings, texts, and Explorables. The position of the content must be adjustable. Furthermore, admins and managers should be able to show or hide content from members. By default, content is hidden. Content can only be made visible after it has been configured. Thus, empty content can not be made visible. Managers and admins are warned by clear signs for empty content.

2.1.3. Non-Functional Requirements

The following list outlines requirements being non-functional. These requirements do not describe functional features of the software project but defines additional constraints for the software project. They include, for example, requirements on interaction with other systems (interface requirements), requirements on maintainability, reusability, reliability, or security (quality requirements), requirements on the interface design and user experience of the software (usability/quality-in-use requirements), or requirements related to human interaction with the software system, such as performance, effectiveness, efficiency, or satisfaction (human factors requirements). [iee18]

First of all, WebWriter currently uses a PocketBase instance in its ecosystem. The existing instance should be used and extended in order to implement the functions and features of the software project.

The software project should be a publicly accessible web application. The platform shall provide a clear and well-structured user interface tailored to the needs of its stakeholders. Actions shall be presented in a way that makes them easily identifiable and understandable for the user. Functions that cannot be undone shall require explicit user confirmation before execution. Tooltips or text labels in general can be used to explain actions and support user understanding. Icons used to represent actions shall be supplemented with text labels or tooltips to ensure consistent user comprehension.

Explorables have three types of access: private, community, and public. Visitors (users not being logged in) can only see and access public Explorables. Logged in users can see community and public Explorables and their own ones (access type private). In terms of general security and data privacy, the access to data should be only granted to eligible people. Therefore, checks for access rights must be added in PocketBase.

Considering reusability and maintainability, the project should implement a modular structure allowing the implementation and reuse of components. New features should be implemented without the necessity of adjusting other modules or components. A modern web framework (e.g. React, Vue, or similar) should be used to ensure reusability and maintainability. The web framework shall allow the implementation of the project

as a single-page application. The single-page application should be indexable by search engines. The project shall be implemented as a monolithic application rather than distributed microservices [27]. Monolithic means that there is only a single application containing all the code. This makes them easier in concerns of deployment, development, testing, and debugging. However, they are not as scalable, reliable, or flexible as microservices. With microservices, a software application is split into smaller services, that can be developed, tested, deployed, and scaled independently.

2.2. Usage in Learning Environments

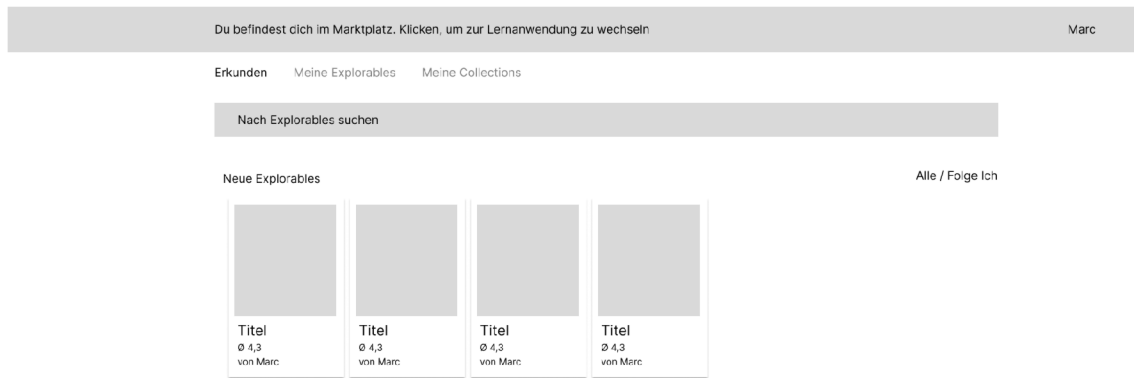
In general, schools and other institutions should be able to use and work with the exchange and learning platform. For a successful integration into schools, I spoke to Simon Frank. Simon Frank is IT system administrator at the Goethe Gymnasium in Stolberg (NRW, Germany) and responsible for a working IT infrastructure for 800 students and 80 teachers. His practical experience in school IT management and support makes him an expert in this field.

Especially management of student accounts and passwords is a problem in schools. [Jü23] When a student forgets its password, the lesson is interrupted: The teacher must reset his password, before the lesson can continue. Furthermore, the platform is designed to be open: Anyone can use the software, whether they are a teacher, student, or someone else entirely. Giving every group administrator the possibility to reset a group member's password won't work because of security risks: As an attacker I can simply create a group, assign another user to that group and reset it's password.

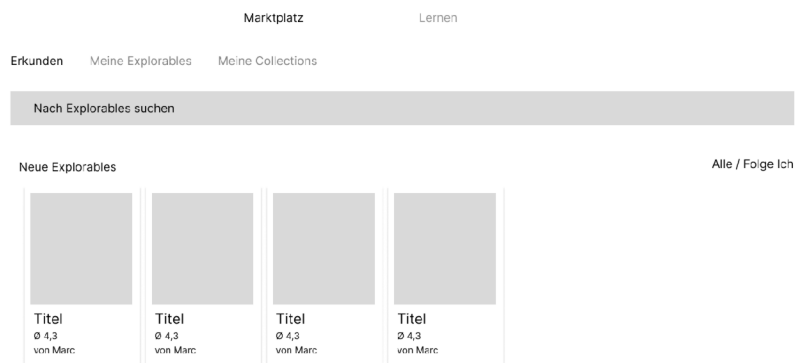
In a first discussion on this problem, we thought about students access using a QR-code. A teacher presents the QR-code and students can join the group in the lesson. It soon turned out that this is not a solution to the first problem but only generates other problems: First, students can only access the learning group and its content, when a teacher shares the QR-code. Students can not work with the group later on, for example at home, when they do not have the QR-code. Second, we thought about using the learning group system also to hand in solutions later on: Therefore, all users are supposed to have an account. This furthermore enables possibilities for later analytics or similar. Third, the contents of a learning group are only accessible to those users that are eligible to access the contents.

After requiring an account for all users, we discussed on how students log in to the platform. Modern login strategies such as passwordless authentication [36] don't require the user to provide a password for his account on the platform itself. The user provides his email address in the login form and then a numeric code (OTP, one time password) or a link (magic link) is sent to his e-mail account. The code must be entered in the login form or the link has to be clicked to successfully authenticate within the application. However, the user then requires it's password for his e-mail account. Furthermore, due to spam filter mechanisms it might be possible that such an e-mail might not be delivered correctly. An alternative to passwordless login by email is passwordless login by SMS. But in order to use that mechanism, the user must provide its phone number and must have access to his phone during the lesson. Both requirements are not very practicable in schools, where phones are forbidden to be used.

2.3. Prototype Design



(a) Navigation using banner



(b) Navigation using menu links

Figure 2.1.: Figma prototypes for the software project. The difference between these prototypes is the navigation: (a) uses a banner, while (b) uses two menu links to switch between exchange and learning platform.

The solution, which I will follow with the implementation of the thesis, is to provide a set of login strategies: The first one is authentication by email and password. Beside this, there will also be a social login: Social login is a strategy, where users can authenticate using existing accounts on so called identity providers such as Microsoft, Apple, GitHub, Google, and so on.

2.3. Prototype Design

The concept as well as the implementation were discussed iteratively in regular meetings with my thesis supervisor Frederic Salmen and Sergej Goerzen. The iterative approach allows the implementation of a feature as well as timely feedback. Using the feedback, adjustments can be made, before implementing the next feature. [SDW⁺10]

A first design prototype was developed using Figma [18]. Figma is a software, where design prototypes and adjustments to these can be implemented faster compared to adjustments made in code or similar. Goal of the first prototype was to discover and discuss the navigation between the exchange and the learning platform. Figure 2.1 shows both of the prototypes.

The first option (a) uses a banner at the top of the page to display, in which type (exchange or learn) of the platform the user is currently navigating. By clicking on the banner, the user can switch the platform type - from learning platform to exchange platform or vice versa. The idea and advantage of this solution is that a user can clearly see in which type of platform he is currently in. He clearly sees what to do to switch the platform type and the platforms are separated from each other. Especially for students, who - in my first understanding - were supposed to only work in the learning platform, this approach was favorable because the students do not see anything from the exchange platform but only the banner. The sub navigation in the prototype (consisting of items "Erkunden", "Meine Explorables", and "Meine Collections") would be only visible in the exchange platform. However, there were two reasons to not further follow this approach: Students are supposed to have the possibility to access the exchange platform on their own and users are not able to see the navigation of the other platform type. If a user, for example, is currently active in the learning platform, he can not see the navigation of the exchange platform. Splitting this up could be confusing to users.

The second prototype, however, was not better for the same reason: Instead of using a banner as proposed in (a), the prototype in (b) uses menu links to switch between the platform types. A user can clearly see the currently active platform type and all available platform types, but they could have a sub navigation. That navigation will also be hidden from the user, if he is not actively navigating in the corresponding platform type. In the final implementation, a single navigation will be used with all navigation links from both platform types.

2.3.1. Inspired by existing software

The two presented prototypes are the only ones that were created for the software project. I first wanted to create a prototype for the whole software. I then realized that a lot of the functional requirements are already implemented in other, existing software: The exchange platform itself is similar to a marketplace such as Amazon [4] or eduki [15]. Collections are similar to playlists on YouTube [60]. And the learning platform is like Moodle [29].

As the features are already implemented in other software, I stopped working on any further prototypes. In this case, prototypes would not add much value, especially compared to the time they require for creation. Existing software is used as inspiration for the implementation of the features and the basic design language is already given by the prototypes. Using the inspiration and the prototypes I started implementing the software project of this thesis.

Chapter 3 Implementation

Based on the concept created in the previous chapter, the following chapter focuses on the realization and implementation of the exchange and learning platform. To first get an overview on all the existing system components, the overall system architecture is presented. After that, an appropriate technology stack is selected with which the actual implementation can be realized. Based on that, functionalities are implemented as well as adjustments to the database model are made. Details on the implementation are presented and decisions and challenges are explained.

3.1. System Architecture

Before diving into implementation details, I am going to present the overall system architecture. The system architecture of this platform depends on the existing system architecture of the WebWriter ecosystem that was already presented in Figure 1.1 - including the platform that is implemented within this thesis. The platform is only meant to access data using the API, which is based on PocketBase. Interaction to any other component of the ecosystem is not required and therefore not implemented.

The software architecture only consists of the platform and the external API. The platform only consists of a single application, which includes server-side rendering (SSR) and client side rendering. I will explain the meaning and importance of server- and client-side rendering in the next section. The application uses an API-Service for the communication with the API that is provided by PocketBase. The communication is based on HTTP

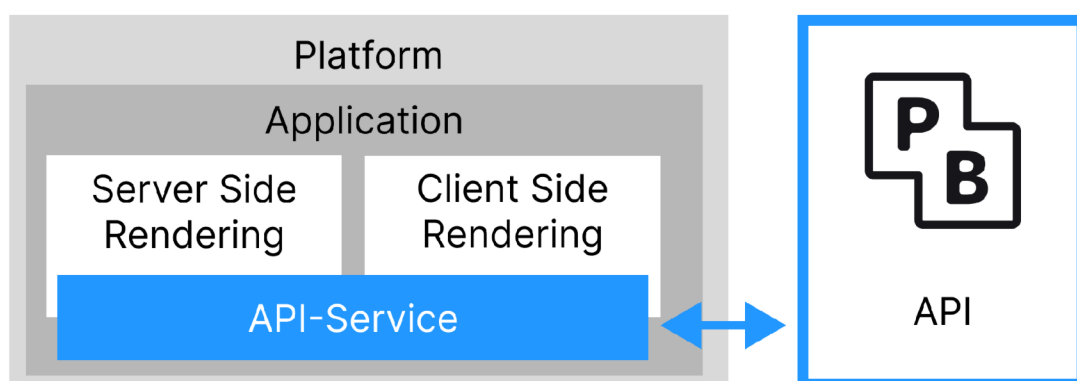


Figure 3.1.: Architecture of the software project. Communication between API-Service and external API is implemented using HTTP/REST.

using REST. Figure 3.1 visualizes the basic software architecture. The system architecture already visualizes the usage of server- and client side rendering. The requirement of using and implementing server- and client side rendering arises from the chosen technology stack.

3.2. Technology Stack

Selecting an appropriate technology stack is an important step in software development. The chosen technologies must be able to fulfill the functional and non-functional requirements defined during the requirements analysis. The decision can be challenging: Although there are a lot of frameworks and libraries available in web development, not every technology or library is suitable for the given use case. Furthermore, some technologies were not allowed by the thesis description.

Additionally, the already existing ecosystem of WebWriter influences the decision: Reusing technologies from the existing ecosystem can simplify maintainability and extendability of the software project itself and for other parts of the ecosystem.

In web technologies, there are a lot of possibilities to implement various types of web applications. Different programming languages such as Java, Go, Python, or JavaScript can be used. Each of these programming languages offer a variety of libraries, frameworks, or other tools. The following discussion focuses on the subset of technologies that are used within this thesis project and the WebWriter ecosystem.

As WebWriter already uses JavaScript, JavaScript and its typed version TypeScript were chosen as the primary programming languages for the implementation of this thesis project.

In the following, the chosen technologies are outlined and the reasons for their selection are explained.

3.2.1. Web Frameworks

The first category of technologies used in this project are web frameworks.

In web development, there are numerous frameworks that can be used to implement application projects such as the one in this thesis. WebWriter itself uses Lit[24]. Technically, Lit is not a framework but a small library that allows the implementation of so called Web Components. For WebWriter, this enables the development of standalone widgets that can be used as custom HTML elements. Using Lit for the implementation of this thesis project could be beneficial in terms of maintainability and extendability of the overall WebWriter ecosystem. However, Lit does not provide certain essential functionalities such as routing and server-side rendering¹. While these missing features could be implemented using additional libraries, the required effort and potential for errors would be excessive in relation to the scope of this thesis. Moreover, such an implementation would also increase the complexity of the software project and the maintenance effort for future

¹ Server-side rendering is currently an experimental feature. More details on <https://lit.dev/docs/ssr/overview/>. Accessed: 15/10/2025

developers.

Considering other solutions is therefore mandatory to ensure a maintainable and stable software application. Popular alternatives to Lit are for example Angular, React, or Vue. Each of them have their own advantages and disadvantages.

Starting with Angular, it has - compared to the other two frameworks - the largest file sizes (also called bundle sizes). Larger bundles increase download times. This also increases the time required until the application is loaded and a user can interact. Generally, Angular provides lower performance in larger applications compared to React or Vue because of its rendering technique. In addition, it is the least preferred framework among the three. Angular has a notably steep learning curve. Initial familiarization for people without prior Angular experience can therefore take a considerable time. [6]

From my personal experience, I can confirm the steep learning curve. In a previous large project where Angular was used, the increasing number of components made the project structure become more and more confusing and unclear. It was difficult to maintain a clear and consistent structure. However, I would like to point out that this might not be solely due to Angular.

The second alternative among the considered web frameworks is React. According to its official website, React is technically a library and not a framework [39]. React produces smaller bundles and has a smaller learning curve compared to Angular. Furthermore, React is currently the most popular of those three frameworks. React benefits from a large community and a wide range of additional libraries. [6]

However, React was not used due to personal experience and preferences. I have not used React before and therefore I have no practical experience with it. This leads me to Vue, which is a framework I have already used before.

Vue.js - or also called Vue - is a framework with average popularity. Its popularity is between that of Angular and React. Vue has the smallest bundle file sizes in comparison to the other two frameworks, is easier to learn and has an active community, which also provides a wide range of libraries. Technically, React and Vue are quite similar in many aspects such as architecture, rendering, or DOM (Document Object Model). Considering these aspects as well as my personal experience and preferences, Vue was chosen as the framework for this thesis. [6]

3.2.2. Server-Side Rendering

Vue provides a solid foundation for building and implementing web applications. By default, Vue produces and manipulates output directly in the browser of a user (client-side rendering). The opposite of client-side rendering is server-side rendering. Server-side rendering (SSR) is a process which allows the rendering of an application on the server. The generated static output on the server is sent to the client, which then processes the static output and turns it into an interactable application. This process is called client hydration. [53]

The approach of using server-side rendering provides several advantages compared to client-side rendering alone. First, it improves the time-to-content, meaning that users can see content on their screen earlier. Especially users with slow internet connection or slow devices benefit from that approach, as they do not have to wait for the application bundle to be downloaded and executed before any output is shown. Instead, the server sends a fully rendered version of the web application which is shown to the user while the JavaScript is downloaded and executed in the background. [53]

Second, SSR is beneficial for search engine optimization (SEO). Search engines navigate through the web and crawl for web page contents. While crawling, the content of a web page is stored and indexed inside a large database of the search engine. When a user performs a web search, the database returns results that best match the search string. Using server-side rendering, the output of the server can be used by the search engine while crawling. Without server-side rendering a search engine would only crawl an almost empty page. Search engines can run JavaScript, but they do not wait for asynchronous calls such as data fetching from APIs or databases. [59] [53]

Vue itself defines a library for server-side rendering which must be integrated into a backend server. However, the official Vue documentation recommends to not implement server-side rendering using the existing library on its own: "A complete implementation would be quite complex and depends on the build toolchain you have chosen to work with" [53]. The official documentation only covers server-side rendering for the basics of Vue. Anything more advanced (routing, data fetching, state management) must be carefully considered and requires additional effort. Because of maintenance, extendibility, and stability, it is not advisable to use it in this application. The documentation recommends the usage of a higher level solution like Nuxt. [53]

3.2.3. Nuxt

Nuxt is a framework built on top of Vue. Besides the integration of server-side rendering it provides functionalities and support for example for routing, data fetching, error handling, search engine optimization, and modules. Modules allow the extension of Nuxt's basic features and therefore implement additional functionalities. Furthermore, Nuxt defines a clear, common directory structure. The common directory structure improves especially the maintainability of Nuxt applications. [31]

3.2.4. PocketBase

While the frontend application is implemented using Vue/Nuxt, the platform also requires a backend system for storing and retrieving data.

For this purpose, the existing PocketBase instance from the WebWriter ecosystem is used. PocketBase provides a performant database including schema builder, validation mechanisms based on defined schemas, and a general-purpose REST API. It also supports user management and authentication via email and password as well as social logins. PocketBase organizes data into collections, each of which can be configured individually. The schema of each collection can be defined using the schema builder. Configurable API rules allow control over data access. The data stored in PocketBase can be accessed using a library provided by the PocketBase developers. [37]

The decision to reuse the existing PocketBase instance was given from the beginning. However, this decision provides clear advantages: it ensures consistency and seamless integration within the existing WebWriter ecosystem and reduces future integration and maintenance effort.

Other tools

Two other tools, that are used within the implementation of this thesis, are worth mentioning: The first of these is GitHub, where the source code is stored and versioned. For this project, a new repository was created. Beside storing the source code, GitHub allows to create and track bug reports and feature requests.

A last tool that was used for the thesis implementation is Docker. "Docker helps developers build, share, run, and verify applications anywhere — without tedious environment configuration or management." [13] The application is being built as a so-called Docker image. An image includes everything that is needed to run a specific application. An image can run on any host machine and does not depend on the underlying operating system. The image is running inside a container on a host machine, which is a process "isolated from all other processes on that host machine" [14]. Every container runs its own software with its own configuration. [13] [14]

3.3. Data Model and Integration

The existing PocketBase instance initially defined two collections: *users* and *documents*. The *users* collection stores general user-related information such as the user's name or the email address. The *documents* collection contains a file reference, an owner field and an access type. The owner field is a relation to an entry in the *users* collection. Every document has exactly one owner. The access type specifies the document's visibility and can be one of *public*, *community*, or *private*. Documents with access type *private* are only accessible by their owner. While documents with community access are accessible to users with an account, public documents are accessible for everyone.

Based on the initial data structure, the database model was extended to support the new features required by the thesis project. Figure 3.2 visualizes the database model that was developed during the thesis. The existing database model consisting of *users* and *documents* was extended by the collections *collections*, *user_profiles*, *user_profile_follows*, *document_reviews*, *documents*, *document metas*, *sources*, *groups*, and *group_content*.

In the beginning, public profiles were implemented. Public user profiles are stored in the *user_profiles* collection. They include all information of a user that should be publicly available. Each user has exactly one profile. Currently, only the profile name and an avatar are stored, although the avatar is not yet used.

The collection *user_profile_follows* was added to store, who (*profile*) is following whom (*follows*). Follower relationships between users are asymmetric: User A can follow user B without the need of B also following A. A user can follow multiple users and multiple users can follow the same one.

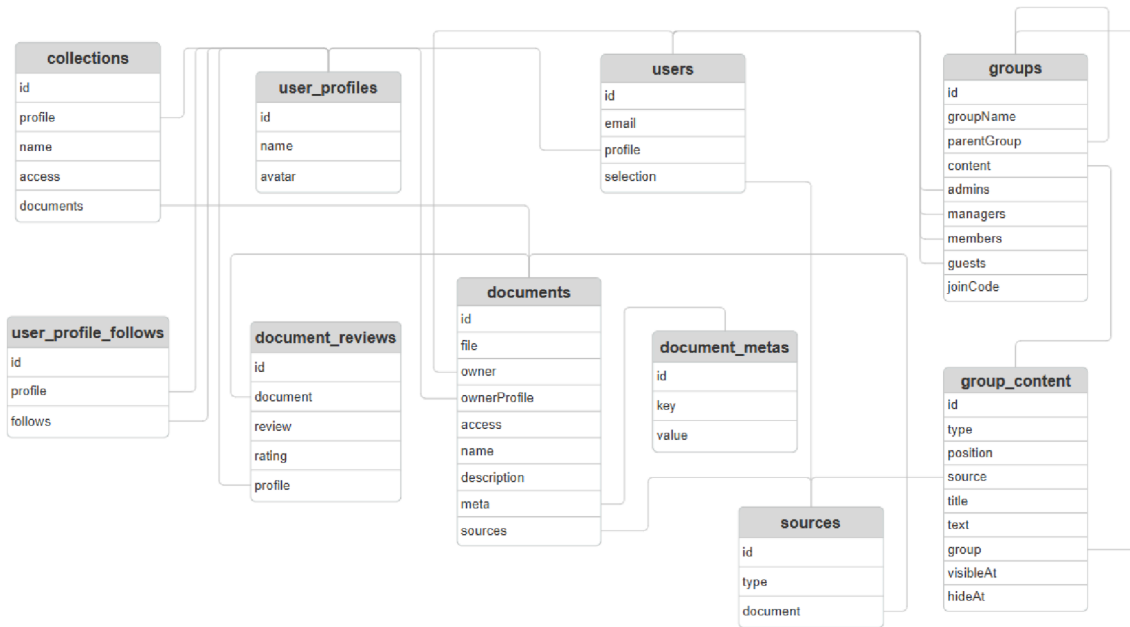


Figure 3.2.: Database Model/Schema implemented in PocketBase

The *collections* collection allows users to store and organize documents for later access. A user can have multiple collections, each being able to store multiple documents. Collections have a name as well as an access type to allow the sharing of collections with other users. Currently, access can only be private, community, or public. Sharing with single, specific users is not possible.

Documents are stored within the *documents* collection. A document consists of a file, an owner and its profile, the access type, a name, an optional description, optional meta information, and optional sources. Meta information includes additional information of Explorables such as authors, license, or keywords. For a single document, multiple meta information can be stored. In the context of documents, sources are references to files, which have been used to compose the Explorable. They are currently not being used but only defined within the conceptual model of this database.

document_metas is a collection that stores key-value-pairs of meta information. Examples for valid keys are "author", "license", or "keywords". Value can be any string.

Another collection related to documents is the *document_reviews* collection. It stores reviews of a user for a specific document. A review consists of a rating, which is a required integer between 1 and 5, and a textual review, which is optional. Using the optional text, a user can discuss and provide detailed feedback for the document and explain the given rating.

For the implementation of the learning application a recursive approach was used: In the *groups* collection all groups are stored. A group can have a parent group. Using the parent group reference, groups can be nested: If one or more groups have a parent group, this also means that each parent group has one or more child groups. This creates a tree structure of groups. Groups without a parent group are the root in the tree structure. This recursive group structure enables hierarchical organization of learning groups and

content and flexible assignment of user roles. Groups furthermore have a name, a list of content, and a list of admins, managers, members, and guests. Admins, managers, members, and guests are lists that include group members. The list assigns a user to a special role of that group.²

The content of a group is stored inside the *group_content* collection. The group content is defined by its type. Currently, three types are implemented: source, title, and text. Furthermore, a relation to the corresponding group as well as the position of the content within this group is stored. To control the visibility of content to group members, dates are stored defining the times, when content becomes visible to or will be hidden from members.

Sources are being used as references for example in documents or for group content. Sources being stored within the *sources* collection are defined by a type. Within this thesis, the only source type is document, which references to a document being stored inside the *documents* collection. Later on, there could also be other source types such as images, videos, or external links.

The data model was extended and reworked multiple times during the implementation. The presented data model is the current version of the data model. Collections or fields managed by PocketBase (e.g. create and update dates for entries) are not presented in the diagram in Figure 3.2. They are automatically generated and managed by PocketBase and therefore not directly related to the development of the data model required for the platform implementation.

PocketBase can be extended using JavaScript. This provides more customization possibilities to PocketBase. For this, PocketBase defines various types of interaction possibilities. In the implementation, Pocketbase was customized using event hooks and routing. Using routing, custom API routes were implemented. They were used, when the built-in API can not be used anymore because of their complexity, logic, or access controls (API rules). Using event hooks, PocketBase can be extended by functions that are executed, when something happens: This could be for example when an API request or a database operation is performed. [17]

The exact use cases for customization in PocketBase will be presented in section 3.4.3. Before that, the project structure and its key components will be presented.

3.4. Implementation Details

The implementation of this thesis project can be evaluated in terms of multiple aspects. The scope of the project includes general details such as the project structure or the usage of reusable components, as well as more detailed aspects such as the implementation of role-based access management in groups. Additionally, it considers potential issues and approaches for improvement.

² The implementation of role-based access in groups is described in section 3.4.4

3.4.1. Project and Folder Structure

The project structure was mostly given by Nuxt. Nuxt is only the framework that is used for a single application. For reasons of clarity, the root folder of the project consists of only a few files: A `package.json` and its associated `package-lock.json`, two Docker configuration files (`docker-compose.yml` and `docker-compose.dev.yml`), a `.gitignore` to define ignored directories and files for git, and an example configuration file for environmental variables (`.env.example`). Beside these files there exists an `services` directory, which includes all mandatory services for the exchange platform: `exchange-platform` itself and `pocketbase`.

Another service is `Explorable-snap`. Although this service was added during the implementation of this thesis project, the service is not part of my implementation. The service, developed by Sergej Goerzen, is a small tool that provides functionalities to generate preview images of Explorables. However, these preview images are currently not being used in the frontend application.

The `pocketbase-service` only contains a `Dockerfile` and a folder containing PocketBase hooks that have been implemented. The `Dockerfile` was copied from the official documentation of PocketBase [19].

The most important service is the `exchange-platform` service. It contains the whole frontend application. Before delving into the application itself, I will describe the development and deployment using Docker. The exchange platform uses two Docker image build files: `Dockerfile_Dev` and `Dockerfile`. The latter one creates an image that is supposed to be used in production environments, for example when deploying it on a server for public access. It only includes the files, that are required to run the application. The `Dockerfile` is automatically used, when using the `docker-compose.yml` in the root directory of the project.

For development purposes, the `Dockerfile_Dev` should be used together with the `docker-compose.dev.yml` from the root directory. The development image includes all files of the project unless they are explicitly ignored (configurable using the `.dockerignore` file inside the service). The `docker-compose.dev.yml` applies a special configuration such that it pushes changes to files into the image. When files in the image change, the development server is restarted by Nuxt automatically. For simplicity, a developer must only execute the `docker-dev` command that is defined inside the `package.json` in the root directory using `npm run docker-dev`.

All other files and folders existing in the `exchange-platform` service are related to Nuxt. Enumerating and explaining all of them is not very practical. The most relevant directories are `components` and `pages`. Inside the `pages` directory, a file and directory structure is implemented that represents the actual path structure of the frontend application. If, for example, the request path is `/my/collections`, the corresponding file for that page in the directory structure is `pages/my/collections.vue`. The directory structure matching the path structure of the application is a feature provided by Nuxt, called file-based routing [35]. Each route has therefore its own Vue component responsible for the page. The file or directory names can also include dynamic route parameters that are written in squared brackets. Dynamic route parameters are used for example for unique identifiers, which may influence the content of the page. This is

used, for example, for learning groups, which are identified by their unique identifier. Based on the unique identifier the corresponding group and its content are shown. Other examples within the scope of the thesis project are detail pages for Explorables or user profiles, where the content is also identified by unique identifiers.

The pages itself are responsible for loading, managing and embedding the content. However, some content is used several times on different pages. Instead of implementing the same content and functions multiple times, which increases maintenance effort, reusable components are used. They are organized in the `components` directory. These components can be used as HTML elements in other components (such as pages). They implement their own logic encapsulated from other components. Passing data into or receiving updates from a component is possible using so-called props and emits. They enable possibilities of communication between components.

Besides the pages and components that have been implemented for this thesis, external libraries may also define reusable components. The frontend application uses multiple external libraries to achieve different functionalities: translation and internationalization, state management as well as icons and style components are only some examples.

3.4.2. External Libraries and Components

Even though Nuxt was used instead of Lit, external libraries used by WebWriter can still be considered. WebWriter, for example, uses the component library Shoelace for the layout and basic design of its interface. Shoelace provides user interface components such as navigation menus, buttons, popups and dialogs, input fields, and many more. Shoelace components are implemented as web components, which are different from Vue components. Vue components are designed to work within a Vue application. Web components, on the other hand, are registered in the browser as custom HTML elements and can therefore be used like native HTML elements. Shoelace also supports integration within Vue applications. [43]

However, the official documentation of Shoelace currently does not list support for Nuxt-based projects [43]. Although Vue is supported, the usage of Shoelace in Nuxt-based applications may lead to technical limitations during server-side rendering. To avoid potential difficulties and ensure consistent rendering on the server- as well as on the client-side, PrimeVue was used instead. PrimeVue is - similar to Shoelace - a component library, which is compatible with Vue and Nuxt. It includes a set of more than 80 styled UI components. [38]

For making the application accessible to more users, internationalization was added. The Nuxt module `nuxt-i18n-micro` [33] was chosen for translating the user interface into multiple languages. Compared to `nuxt-i18n`, another internationalization module, `nuxt-i18n-micro` produces smaller bundles, requires less memory, and has a higher performance in regards of build time and server performance. Currently, German is the only supported language and not all texts are translated, but the technical foundations are given.

The next section covers the integration of PocketBase into the frontend application. The integration of PocketBase is based on so-called stores. The Pinia library was used for this purpose, which is also officially used in Nuxt's documentation for state management. Pinia stores are globally accessible and can therefore be used in any component of the frontend application. They can store data (so-called state) but also provide actions. [45]

3.4.3. Integration of PocketBase in the Application

Using the stores of Pinia, a store was implemented to manage and integrate the PocketBase JavaScript SDK into the frontend application. In Figure 3.1, where the architecture of the software project was presented, this store is defined as "API-Service". For the implementation of the store, composables were used. Composables are functions that "encapsulate and reuse stateful logic" [9].

The state of the store especially saves information about the current authentication status: Is a user authenticated, which user is authenticated, and what is its authentication token. The authentication token is for example required when calling custom API routes that are protected by authentication. Furthermore, the store defines actions. These actions can be used to query, insert, update, and delete data via the PocketBase API. They provide a common interface that can be used in any of the project components. Identical requests to PocketBase do not have to be implemented multiple times in different components. If changes are made to PocketBase and the corresponding code needs to be adapted, this now only has to be done in one place. Overall, this makes the code less redundant, less error-prone and easier to maintain.

By default, PocketBase uses the browsers local storage for storing authentication tokens [25]. If a request to the PocketBase API is performed by the client, the JavaScript SDK reads the authentication token and adds its value to the request [8]. Using the authentication token, PocketBase can authorize and assign a user to the corresponding request. However, the browsers local storage is not available in the server-side rendering context.

In concerns of search engine optimization, this behaviour will not be a huge problem: Publicly available information will be still accessible for search engines as it does not require authentication. Pages and components being inaccessible for unauthorized users will in general be also inaccessible for search engines.

Server-side rendering is not only used by search engines but also by normal users on their initial request. If a user is authenticated in the browser but not in the context of server-side rendering, the server will respond with the warning, that the user is not authorized to access the requested content until the client-side hydration finishes and renders the requested content. Here, content may flash and cause unexpected layout changes: The user may first see a warning that is then replaced by the actual content. This is called a hydration mismatch. Hydration mismatches increase the time until the application gets interactive: If a hydration mismatch is detected, the entire component tree is re-rendered by Vue. Due to flashing contents and layout shifts, this may furthermore decrease the users experience with the application. [32]

PocketBase does not support and also warns about using cookie-based authentication with PocketBase. PocketBase recommends to use other backends than PocketBase, if Nuxt or similar frameworks are being used. Because of the existing PocketBase instance in the WebWriter ecosystem, this was never a solution to the problem. Using cookie-based authentication as well as PocketBase within a server-based context may introduce a security vulnerability: PocketBase client instances as well as the corresponding authentication store may be shared over multiple requests from different users. If a first user authenticates in a request and another unauthorized user also performs a request, the request of the second user may be successfully be authorized by the first users authorization token. [22]

Because of the warnings and the possible security vulnerability, an alternative solution was discussed. PocketBase allows user impersonation [21]. User Impersonation means, that another user sends requests as another user. In PocketBase, so called superusers are able to impersonate other users. The superuser will access data in PocketBase in favour of another user. Due to security concerns, the impersonation must be integrated in the backend of the Nuxt application and the client will request data using Nuxt as a middleware between the client and PocketBase. This furthermore requires another authentication layer (e.g. session cookies) between the client and the Nuxt application, because the impersonation happens between the Nuxt application and PocketBase. For storing the relational assignment between session cookies and the impersonation another structure - for example an external database or a cache - must be used.

In conclusion, the approach of using user impersonation seems to be too complex to implement and maintain. It furthermore seems to also introduce security vulnerabilities: A single superuser can interact as any other user existing in PocketBase. Although the approach of implementing cookie-based authentication with PocketBase was not further considered due to security vulnerabilities, it was the approach that was actually implemented in the end.

PocketBase provides some information and basic implementations for cookie-based authentication, also for Nuxt [44]. The implementation consists of two components: The first one is the `CookieAuthStore`, which is an extension of the `BaseAuthStore` that is implemented in the JavaScript SDK of PocketBase. The second component is a composable handling the usage of PocketBase clients. The latter one requires the `CookieAuthStore`.

Both implementations depend on different methods based on the current rendering context. The `CookieAuthStore` (see Listing C.1) implements reading and saving cookies through the browser as well as the server. Based on the context, different methods must be used: On the server, the cookie read from the request headers while the browser implementation uses Nuxts composable `useCookie` to read the cookie. Using the function `loadFromCookie` defined in the PocketBase JavaScript SDK, the authentication store is loaded from the corresponding cookie. If the authentication store changes, the update is written to the cookie store in the browser. For the browser implementation, this is done using the `useCookie` composable again. In the server implementation, the cookie is set as a header within the request's response.

The composable `usePocketbaseClient` (see Listing C.2) also differentiates between server- and client-side rendering contexts: To mitigate the security vulnerability of shared authentication stores between multiple requests, the composable creates a new PocketBase client every time the composable is used on the server-side. On the client-side, the PocketBase client is reused. If no PocketBase client exists, a new one will be created and stored for later usage. When creating the PocketBase client, the `CookieAuthStore` is initialized and passed as the authentication store to be used. Furthermore, automatic cancellation of requests is disabled. Otherwise PocketBase would cancel same requests initiated by different components and only the last request would return the requested data. Because this was a frequent problem while implementing, auto cancellation was turned off.

The final implementation uses `usePocketbaseClient` in every component, where PocketBase is used in the component. The call of the composable happens in a synchronous context of the component directly within the `setup` function. If the composable is called within asynchronous contexts, Nuxt may return undefined when calling `tryUseNuxtApp` on the server-side [34]. The `setup` function is the entry point for components being used by Vue [10]. The PocketBase client being returned by the `usePocketbaseClient` must then be passed to the actions of the PocketBase store.

But not only the integration of PocketBase into the frontend application was implemented but also PocketBase itself was extended by implementing custom event hooks and providing further settings. The configuration for the integration of social logins can be done using the administration interface provided by PocketBase. External providers can be added inside the `users` collection using OAuth 2. For testing purposes, social logins using Google and Microsoft accounts were added. The login page of the frontend application reads the list of allowed social logins from the PocketBase API dynamically. One of the hooks implemented is responsible for creating an entry in the `sources` collection when a document is added such that every document has a unique source id, which can be referenced for example in group contents. Another example includes an API endpoint in PocketBase to return all documents of users that the authenticated user is following.

For managing groups, four more hooks were implemented. One for adding members to groups by group admins or group managers. One for users to join a group using a join link or QR code of the group. And two more that are responsible for assigning user roles to groups: The first hook provides an API endpoint for the manual assignment by a group admin or manager. The other hook is called, when a new group is created. It is executed if a request to create a new group is performed. It then assigns user roles according to the intended behaviour of the role-based access system in groups.

3.4.4. Role-Based Access in Groups

A role-based access control system (RBAC) was implemented to manage access and permissions in learning groups. The roles are predefined and currently consist of four roles: administrator (or admin), manager, member, and guest. Based on the role of the user, the user has access to different functionalities in the group. A list of permissions being granted by each role is shown in Table 3.1.

In the requirements analysis only three roles were defined: administrators, managers and members. The guest role was additionally used to grant users access to a group and

Permission	Administrators	Managers	Members	Guests
Can view group	+	+	+	+
Can edit group	+	+		
Can delete group	+			
Can view subgroups	+	+	+	+
Can create subgroups	+	+		
Can view content	+	+	+	
Can create content	+	+		
Can update content	+	+		
Can delete content	+	+		
Can add members to group	+	+		
Can assign administrator role	+			
Can assign manager role	+	+		
Can assign member role	+	+		

Table 3.1.: List of permissions granted by role in a specific group

its subgroups, but not to their contents. This role is only to identify and maintain the access of a user to a group inside a hierarchical tree structure of groups. If a user receives a role, the same role is also applied in subgroups. For parent groups, the user receives the guest role but only if he does not have another role. The guest role therefore allows a user to navigate through the hierarchical structure of groups to access content of groups he is allowed to.

Figure 3.3 shows an example of a possible yet simplified group structure. At the top level there is a school, which organizes its learning groups by terms. For the simplified example, the terms "2023/2024" and "2024/2025" were created. Each term includes all the possible classes, reduced to classes "7a" and "7b" in term "2023/2024" as well as the consecutive classes "8a" and "8b" for the "2024/2025" term. The class "7b" also has two courses visualized as groups: "German" and "English".

Before the user has been assigned to one of the roles to any of these groups, he can not see any of them. Now imagine that the user is assigned the manager role to the "2023/2024" group. He then receives the guest role in the "School" group as well as the manager role in the following groups: "2023/2024", "7a" and "7b", "German" and "English". As the user has no role assignments in the right part of the tree, no roles will be assigned in the groups "2024/2025" nor "8a" nor "8b". Next, the user is promoted from manager to an administrator for the "German" course. He is assigned the corresponding role in the course and no other role assignments or changes are performed. There are no subgroups the user could be assigned to the administrator role and he is already assigned to roles in the parent groups. Last, the user is assigned to the member role in the class "8b". Again, no assignments to subgroups are performed, because there are no subgroups yet. But, the user is assigned to the guest role for the group representing the term "2024/2025".

If, for example, a new group "Spanish" is created as subgroup of the "8b", the user would automatically be assigned with the member role to that new group. However, the current implementation lacks the possibility to unassign roles of users without generating inconsistent states in the role inheritance between groups. If, for example, a user is assigned from manager to member in the "English" group, the role inheritance from parent groups

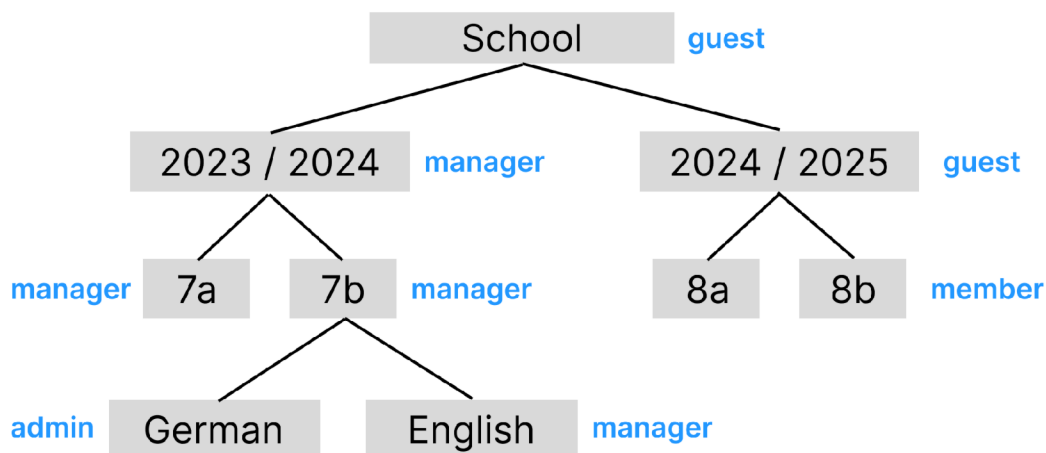


Figure 3.3.: Example visualization of a hierarchical group structure including possible group assignments for a single user

is violated: In the group "7b" the user has the manager role and he should therefore have the same or a higher role in all the existing subgroups.

To encounter this issue, the source group of a role assignment should be stored. A user can then only be removed from a role assignment in the specific group, where he was explicitly assigned to that role. Using the given example, the user could only be removed from its associated roles in the "2023/2024" group, where he was assigned as a manager, the "German" group, where he was assigned as an admin, or the "8b" group, where he was assigned as a member. The role tree could then be calculated and adjusted on updates using the stored role assignments for relevant groups. It then does not matter, if a user is assigned to or unassigned from a role.

During development this approach was already implemented. It turned out, that this approach did not work well together with the PocketBase API rule system. For simplicity reasons, the implementation was removed again and replaced by the current implementation. As both implementations had and have advantages and disadvantages, a combination of both approaches would be advantageous: Use the current system, which works well with the API rules as well as the adjustments for storing the source of a role to encounter problems with the current system.

3.5. Final Result

The previous sections have established the fundamentals and explained the implementation in detail. With this section, the final prototype will be presented, before it is evaluated in the next chapter. I will first provide an example on how to create a page using the previously defined composables, which shows all stored, publicly accessible Explorables. The following code is a reduced excerpt from the `pages/index.vue` file, which actually represents the start page of the exchange platform.

```

1 <template>
  <h2>Erkunden</h2>
  <ExplorableGroup :explorables="documents"></ExplorableGroup>

```

```
</template>
5
<script setup lang="ts">
  import type { Explorable } from '#shared/types/Explorable';
  import ExplorableGroup from '~/components/ExplorableGroup.vue';

10  const pb = usePocketBase();
  const pbClient = usePocketbaseClient();

  let documents: Ref<Explorable[]> = ref([]);

15  const loadDocuments = async () => {
    documents.value = await pb.getDocuments(pbClient);
  };

  await loadDocuments();
20 </script>
```

Listing 3.1: Example implementation of a page using the previously implemented composables. The example is a short version of the main page.

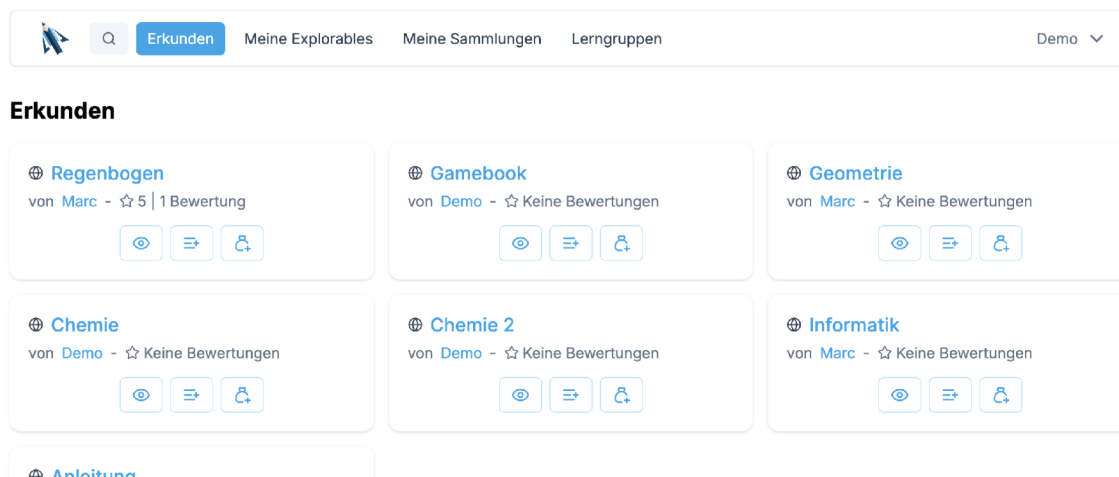
The example represents a Single File Component for Vue. It consists of a template, which represents the markup of the page, and a setup script, which is the logic of the page implemented in TypeScript. The template consists of a title (h2) as well as a reusable component (`ExplorableGroup`). The reusable component is used to show a list of Explorables, which are passed using the `documents` variable. In the setup script, directly after the imports, the two presented composables `usePocketBase` and `usePocketbaseClient` are used, before the `documents` variable is defined. With the function `loadDocuments`, which is called at the end of the setup, Explorables are requested from PocketBase using the corresponding function inside the store implementation. The return value is set to the value of the `documents` variable.

The final result of that rendered page can be seen in Figure 3.4 (a). Each of the Explorables that were loaded from PocketBase is shown as a single card within the list. Here, the cards are also reusable components. Within this small example, the relevance of reusable code becomes visible.

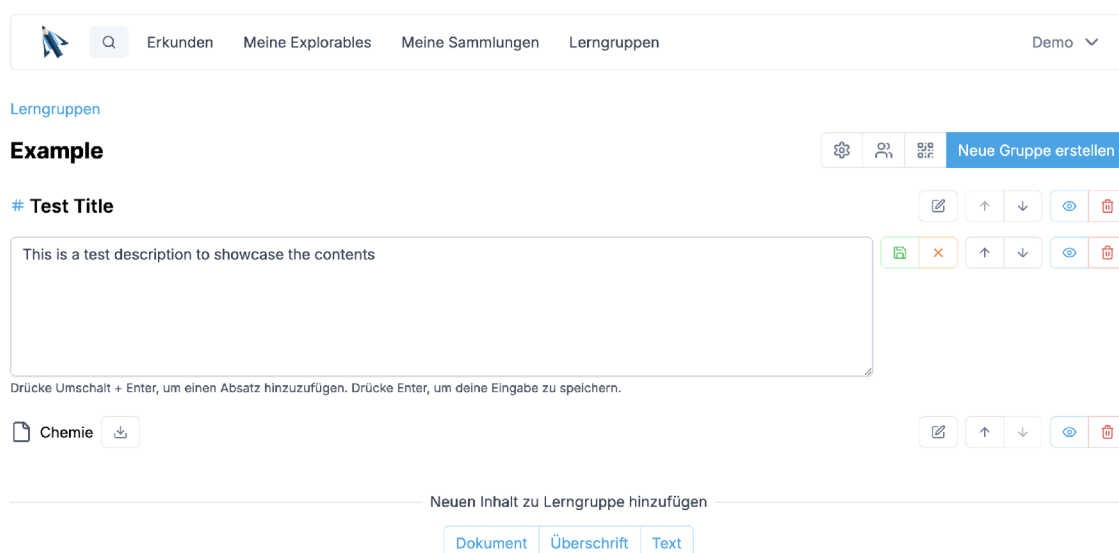
Another example of a page is shown in Figure 3.4 (b). It shows a learning group (called "Example"), where each of the supported group content types are used. For the content type "text" the edit mode is activated. On the right hand side of the content, users find control buttons for toggling the edit mode, changing the position or toggling the visibility of the content within that group as well as deleting the content. At the top of the learning group, buttons can be found to open group settings, manage group members, share the group using an invitation link/QR code, and to create a new subgroup inside the current group.

3.6. Design Decisions, Challenges and Improvements

The implementation was influenced by a variety of design decisions and challenges. In the following, I am going to summarize the most important parts of this implementation once again and also provide some first possible yet basic improvements.



(a) Screenshot of the start page that is shown when accessing the exchange platform.



(b) Screenshot of a learning group with edit mode activated for a single group content type.

Figure 3.4.: Reduced collection of screenshots that show different types of pages.

The most important design decision was related to the choice of the web framework that is being used. Nuxt was used because of its modularity, its various utility functions and the predefined, common directory structure. The directory structure makes the code more maintainable for future development. To provide maintainable code, functions and variables were named clearly. Function names are designed to explain what they do without additional comments. Thus, only more complex functions were provided with more detailed and explaining documentation.

Compared to Vue only, Nuxt provides server-side rendering to achieve a better search-engine optimization as well as a better user experience. Simultaneously, server-side rendering turned out to be one of the more challenging parts. While libraries could be selected based on their compatibility with SSR, the use of PocketBase was predetermined. Integrating PocketBase so that it works well with server-side rendering was challenging. However, during further development, testing and evaluation this integration did not

cause any further problems.

In general, the usage of PocketBase defined some decisions in regard to the data model and the corresponding implementation. A first example of adjustments to the data model in favor of PocketBase are redundant references to other collections. This includes for example the collections `groups` and `group_content`, where the `groups` collection includes the `content` field storing a list of references in `group_content`. On the other hand, `group_content` has the field `group` which references back to a single entry in the `groups` collection. The back-reference is maintained, to simplify the corresponding API rules.

Another example is the creation of the `profiles` collection within PocketBase. The goal was to separate public from private user information. When users are created, especially using social logins, their associated profile is currently not created automatically. The profile is created by the user during the next login. By implementing a corresponding PocketBase hook, the profile creation could be performed automatically when a user is created. However, a more suitable solution might be to completely drop the `profiles` collection. Private information can be hidden in PocketBase so that it is not made public by the API. I would then provide an API route for users that returns the private information for the currently logged-in user. This reduces the complexity of the data model and simplifies the maintainability. Furthermore, I would rework the role-based access control system that is used in groups. It currently lacks some mandatory functionalities and may also be confusing to users.

A final study was designed to test the usability of the implemented platform, identify potential weaknesses and problems, and determine which functions were not considered and could be implemented in the future.

Chapter 4 Evaluation

The following chapter describes the process of designing, realizing, performing, and analyzing the user study, which was used in particular to evaluate the usability of the software. The goal of the study is to test the implementation with users of the target group. The target group can provide valuable feedback in regards of usability, user experience, expected behavior as well as needed functionalities.

This chapter first starts with the theoretical foundations on which the study bases on. Then the design and preparation is presented. The preparation includes a pre-study that was used to test the design before starting the actual study with users from the target group. At the end, observations and results of the study are shown and discussed.

4.1. Theoretical Foundations

For evaluating usability and user experience, multiple different approaches can be used for measurement. First of all, usability is one aspect of user experience. While user experience covers the whole experience of a user before, while and after using a software product, usability itself covers only the experience during the usage of a software product. [50] [51]

One of these measurement approaches is the System Usability Scale (SUS). To measure the usability, the System Usability Scale uses ten statements using which a product can be rated. Each statement is rated within a range between 1 and 5. On that scale, a rating of 1 indicates strong disagreement and a rating of 5 strong agreement with the given statement. All ten statements must be rated. To prevent response biases, statements were alternated between positive and negative. To calculate the overall rating, the rating of each statement is calculated depending on whether they are positive or negative statements. For each statement, a common rating between 0 and 4 is calculated. To get the overall usability score, the sum of all ratings is then multiplied by 2.5. The overall usability score is a number between 0 and 100, where a score likely to 0 means poor and a score likely to 100 good usability. [Bro95]

Another measurement approach is the User Experience Questionnaire (UEQ). It covers the whole user experience instead of only the usability. The UEQ consists of six categories: Attractiveness, Perspicuity, Efficiency, Dependability, Stimulation, and Novelty. Each of them includes four adjective pairs of opposites (e.g. efficient-inefficient, good-bad, complicated-easy, ...). Attractiveness includes six pairs of opposites. The user rates each pair on a scale between 1 and 7. The rating depends on which of the adjectives better describes the user's perception of the product. The results of each category are shifted to

their own scale between -3 and 3. The higher the score, the better is the user experience regarding the corresponding category. For the calculation of the scores, the mean value is used. Therefore, it is unlikely to get scores above 2 or below -2. Simultaneously, a score above 0.8 is interpreted as positive and a score below -0.8 as negative. A score in between -0.8 and 0.8 is interpreted as neutral - neither good nor bad. [LHS08] [52] [Sch23]

To decide for one of these measurement approaches it is important to carefully compare them against each other. Both - the SUS as well as the UEQ - are commonly used to evaluate the user experience and usability of products. The System Usability Scale has 10 statements, which focus on the usability only and not on the user experience. The resulting score rates a product on a single scale. The SUS is not as detailed as the UEQ. The User Experience Questionnaire on the other hand includes the whole user experience instead of only the usability. It evaluates a product using 26 items in 6 dimensions. For a detailed evaluation, the UEQ is the better approach while the SUS can be used for fast usability evaluations. Because of the more detailed results, the UEQ could be the better approach for evaluating the implemented software project, especially in regards to the overall workflow in learning contexts. [54]

Based on the User Experience Questionnaire, the UEQ+ was developed. It is a modular extension of the UEQ. It includes the same categories as the UEQ, but additionally defines other categories. As the UEQ+ is a modular approach, the author can choose those categories, that are relevant for evaluating the product. For the UEQ+, as well as the UEQ, a data analysis sheet is given using which the calculation of the scores can be performed. However, the UEQ+ requires some more effort to set up compared to the UEQ. The UEQ is a standardized test with no effort for adjustments. The UEQ+ on the other hand requires the author to know, which categories should be evaluated for. It furthermore might be challenging to interpret the results of the categories defined by the UEQ+, because the additional categories do not have as much scientific data and experience as the existing ones defined by the UEQ. [ST23] [47]

4.2. Preparation

Although the User Experience Questionnaire might be sufficient for evaluating my thesis project, I decided to use the UEQ+. The UEQ+, however, is only one part of the evaluation. The evaluation includes various tasks. For the study to evaluate the thesis project I decided to start with a survey to collect general information about the participants. Afterwards, participants are supposed to execute a list of tasks, which guide them through the software application. Afterwards, the UEQ+ is used for the user experience evaluation. At the end, a small number of open text questions are asked to provide participants with the opportunity to give more general feedback that is not necessarily related to usability.

The study was designed in German and consists of three steps. The English translations of German tasks will be given in round brackets. The first step is a survey to gather general information about the participants. The general information include demographic characteristics:

- **Age** of the Participant. Selectable values are ranges <30, 30-39, 40-49, 50-59, and >=60

- **Gender** of the Participant. Selectable values are "männlich" (male), "weiblich" (female), "divers" (diverse), and "Sonstiges" (other)
- The **teaching subjects** of the participant
- Current **position** of the participant. Selectable values are "Lehrkraft" (Teacher), "Referendar*in" (Trainee teacher), "Lehramtsstudent*in im Praktikum" (Internship trainee teacher), "Lehramtsstudent*in" (Teacher training student), and "Sonstiges" (other)
- **Type of School**. Selectable values are "Primarstufe / Grundschule" (Elementary school), "Sekundarstufe 1 / Haupt- Mittel- und Realschule und Gesamtschule" (lower secondary school), "Sekundarstufe 2 / Gymnasium" (Upper secondary school), "Sonderschule / Förderschule" (Special school), "Berufsschule / Berufsfachschule" (Vocational school), and "Sonstiges" (other)

Collecting demographic characteristics is important in order to analyze, interpret and discuss the results. These characteristics can also help to identify problem groups as well as groups that have few or no problems using the software. Without these information, a more detailed analysis may not be possible and only general conclusions can be drawn. [Ham11]

Beside the demographic characteristics, which were required to be answered, participants were asked to optionally answer 4 questions in regard of their current usage of digital learning materials:

- "Wie finden Sie derzeit für den Unterricht geeignete Unterrichtsmaterialien?" (How do you currently find teaching materials that are suitable for your lessons?)
- "Greifen Sie dazu auf (kostenpflichtige) Angebote zurück?" (Do you use (paid) services for this purpose?). Participants could select between whether the use paid or free services. They could also answer with both options (paid and free) and were supposed to write down examples for such services.
- "Wie erhalten Schülerinnen und Schüler die von Ihnen für den Unterricht verwendeten Materialien? Greifen Sie dazu auf digitale Lösungen und Plattformen zurück?" (How do students receive the materials you use for teaching? Do you use digital solutions and platforms for this purpose?)
- "Wie häufig lassen Sie Schülerinnen und Schüler interaktive Apps als Teil Ihres Unterrichts nutzen?" (How often do you let students use interactive apps as part of your lessons?). The participants could select between "1-2x täglich" (daily), "1-2x wöchentlich" (weekly), "1-2x monatlich" (monthly), "1-2x jährlich" (yearly), and "gar nicht" (not at all)

Depending on the answer of the last question, another question was added. If the participant selected daily, weekly or monthly usage of digital interactive apps, the following question was asked: "Auf welche Art von digitalen (interaktiven) Unterrichtsmaterialien greifen Sie häufig zurück?" (What kind of digital (interactive) teaching materials do you frequently use?). If the participant uses digital interactive apps only a few times a year or not at all, the following question was added: "Warum setzen Sie seltener auf digitale (interaktive) Unterrichtsmaterialien?" (Why do you use digital (interactive) teaching materials less frequently?). Both questions were open text.

The non-demographic questions were created by thinking about, what kind of information may be important for further analyses and influence development. They help to understand, in which extent participants use digital learning materials and how their workflow is constructed, to provide these learning materials to students.

After the first survey, the participants were provided with a list of tasks for the practical part of the study. An alternative of providing tasks to users and guide them through the application is to send them a link to access the application and let them interact with it freely. That approach was not used, because participants may do not know what they could do in this new, unfamiliar application without a clear goal. Although a single, open goal could be formulated, I thought the approach of providing multiple, small goals could be the better one. In the end, eleven tasks were formulated. Sensitive information was replaced by placeholders in squared brackets.

1. Rufen Sie die Austauschplattform in einem Webbrowser auf (Open the exchange platform in a web browser): [URL to the deployed version of the exchange platform]
2. Melden Sie sich mit folgenden Zugangsdaten in der Plattform an (Login to the platform with the following credentials):
E-Mail-Adresse (E-Mail address): [demo user email]
Passwort (Password): [demo user password]
Hinweis für weitere Aufgaben: Ihr Nutzernamen ist „Demo“ (Note for further tasks: Your username is “Demo”)
3. Öffnen Sie ein Explorable, das nicht von Ihnen ist, und bewerten Sie es. (Open an Explorable that is not yours and rate it.)
4. Öffnen Sie das Profil des Autors des Explorables, das Sie gerade bewertet haben. (Open the profile of the author of the Explorable you just rated.)
5. Folgen Sie dem Profil und speichern Sie mindestens eins der Explorables in einer neuen Sammlung. (Follow the profile and save at least one of the Explorables to a new collection.)
6. Erstellen Sie eine neue Lerngruppe. (Create a new learning group.)
7. Bearbeiten Sie den Titel Ihrer neu erstellen Lerngruppe. (Edit the title of your newly created learning group.)
8. Fügen Sie den Nutzer mit der E-Mail-Adresse [second user email] Ihrer Lerngruppe hinzu. Machen Sie ihn anschließend zum Betreuer der Lerngruppe. (Add the user with the email address [second user email] to your learning group. Then make them the manager of the learning group.)
9. Fügen Sie neuen Inhalt zu Ihrer Lerngruppe hinzu. Probieren Sie ein paar Einstellungen aus. Verwenden Sie dabei jeden Inhaltstyp mindestens ein Mal. (Add new content to your learning group. Try out a few settings. Use each content type at least once.)
10. Machen Sie alle Inhalte für Kursmitglieder sichtbar. (Make all content visible to group members.)

11. Löschen Sie Ihre Lerngruppe wieder. (Delete your learning group again.)

After participants finished these tasks, the second survey started. They were first supposed to answer selected categories of the UEQ+. The survey contains the categories that are also included in the UEQ: Attractiveness, perspicuity, efficiency, dependability, stimulation and novelty. Additionally, these categories were extended by intuitive use, clarity and usefulness. Thus the UEQ is included in the survey as well as additional categories that complete the evaluation of the user interface and the platform itself.

The additional categories - intuitive use, clarity and usefulness - were added, because they provide further insights on the product. Clarity, for example, is a category that was designed for evaluating products with a graphical user interface. For this evaluation, clarity provides insights, whether the implemented user interface is well structured. Intuitive use provides insights, if the platform is applicable for teaching, scenarios without the need of time-consuming training courses before. [47]

The following descriptions of the used categories are taken from the official documentation of the UEQ+ [47]:

- **Efficiency:** Impression that tasks can be finished without unnecessary effort.
- **Usefulness:** Impression that using the product is beneficial.
- **Perspicuity:** Impression that it is easy to learn how to use the product.
- **Dependability:** Impression to be in control of the interaction with the product.
- **Intuitive use:** Impression that the product can be used immediately without any training or help.
- **Stimulation:** Impression that it is interesting and fun to use the product.
- **Novelty:** Impression that the product design or product idea is creative and original.
- **Attractiveness:** Overall impression concerning the product. Do users like or dislike it?
- **Clarity:** Perception that the user interface is well-structured and of low visual complexity.

4.3. Pre-Study

Before the main study was performed, the study was tested. Performing a pre-study - also called pilot testing - helps to try out the study and find any severe problems in advance. In general, such pilot tests are done using a person from the target group. Unfortunately, I did not manage to get that much participants for the main study such that I used a person for performing the pre-study, who is part of the target group. [55]

In general, the pre-study should cover the same procedure as the main study. If the pre-study already works well, the results could also be included in the main study. Since the participant being used for the pre-study is not part of the target group, possible results can and will not be transferred to the main study. I therefore decided to not test the survey itself during the pilot study. Demographic characteristics as well as information about experience and workflow in education can not be answered completely or are not relevant for participants outside of the target group. Furthermore, the UEQ+ is a standardized test.

For the pilot study, only the tasks of the practical part were tested. The original list of tasks should be tested with regard to the following aspects:

- **Feasibility:** Is it possible for participants to understand and perform the tasks?
- **Task Coverage:** Do the selected tasks cover all the required and/or important functionalities?
- **Time Requirements:** How long do participants require to perform the tasks?
- **Technical Stability:** Do problems and errors exist, that hinder participants from performing the tasks?

Those four aspects were the most important one for me. Especially the combination of task coverage and time requirements were a challenge: Are the selected tasks sufficient for participants to get familiar with the application for an adequate evaluation? Can the tasks be completed within a reasonable time limit?

The pre-study initially consisted of eight tasks while the final study consists of eleven tasks. The tasks 6 to 11 of the final study, which cover the learning platform, already existed in the pre-study. Instead of the tasks 1 to 5, the pre-study only consisted of two tasks, which covered opening an Explorable as well as storing the Explorable in a collection. It turned out, that the tasks did sufficiently cover the learning platform, but not the exchange functionalities of the platform. Hence, the task list was extended to include eleven instead of eight tasks. Furthermore, the descriptions of the tasks were reformulated with regard to clarity.

During the pilot testing, some errors have been found. Most of them were problems with the naming or labeling of buttons and other text elements. These texts have been adjusted. Furthermore, a bug was fixed that caused users to not be redirected to the learning group overview when deleting a learning group.

4.4. Main Study

With the results of this pilot study, the main study was prepared and could therefore be performed. For the main study, the application was deployed on a server. Using the backup-functionality of PocketBase I prepared the study, created a backup and restored that backup for every participant such that each of them starts with the same database state. The state only included two users and seven Explorables. The Explorables had a title and an author. No license, reviews, ratings, meta information, or descriptions were given. Furthermore, no collections or learning groups were created.



The screenshot shows a dialog box titled "Teilnehmer" (Members) with a close button (X) in the top right corner. Below the title, there is a text label "E-Mail-Adresse des Nutzers, der neu hinzugefügt werden soll." (Email address of the user to be added). Underneath this label is a text input field. To the right of the input field is a blue button labeled "Hinzufügen" (Add). Below the input field, on the left, is the text "Demo". On the right, there is a dropdown menu with the text "Admin" and a downward arrow.

Figure 4.1.: Members List of a Learning Group

The study was performed using four participants. The participants were invited by email or similar platforms and were therefore not randomly chosen. Each of these sessions were performed remotely using Zoom [61]. After the welcome, participants were asked for their experience with WebWriter. Although WebWriter is not directly required for performing the main study, it might be helpful for all participants to know the basics about WebWriter. All participants were familiar with WebWriter, therefore I sent them a link to the survey.

Participants then executed the first part of the survey, which included the demographic characteristics and general information. After they finished with that part, I sent them a PDF-file which included the tasks. For the practical part, I asked participants to share their screen while solving the tasks. This allowed me to observe their actions and help them, if any problems arise.

During these sessions, various problems were observed. Some of these were related to the implementation of the thesis project (e.g. missing translations or mixing of English and German translations). One mistake, most participants encountered, relates to the task to add a new member to a learning group and assigning the manager role. The corresponding dialog (see Figure 4.1) consists of an input field for an email address of the user to add with an associated "add"-button to perform the action. Below of that a list of current group members is given. On the left, the name of the group member is given as well as a role selector on the right hand side. Participants often used the role selector to select the manager role for a member and therefore unassigned the administrator role from their account. Based on the Gestalt Principles, this problem is mostly related to the principle of proximity: Things that are close together may be related to each other. Adding some space or making the list of users more clear (e.g. using a table with a header row) can help to solve that problem. [1]

Another problem for participants was the task to show all contents to all group members. Showing or hiding group content from group members is performed using a button: The button uses icons to describe the action: An eye to show content and an crossed out eye to hide content. The usage of these icons seems to be counter-intuitive: Participants seemed to be confused about the current state. Although the other buttons (e.g. edit or delete) in that list also describe the action that is performed when clicking the corresponding button, this does not seem to work for toggling the visibility of contents. Instead of the action, that is performed when clicking that button, the current state should be shown.

Although the participants were guided through the application by the tasks, they were not explicitly allowed nor prohibited to try things out. One participant, for example, tested to create subgroups in groups. It turned out, that the labeling to create a subgroup is misleading. Furthermore, the inheritance of role assignments is not clearly visible or recognizable. For this, explanations should be provided.

After the participants finished the practical part, they could continue with the survey, where they answered the UEQ+ and the remaining open questions related to the platform prototype. After they finished with that last part, I thanked the participants for their time and effort before the meeting was quit.

4.5. Results of the Main Study

The feedback given throughout the study is important for evaluating the implemented prototype and influencing future development accordingly. For further development it is important to keep in mind, that the target group is not represented by this study. This study consisted of four participants, each of them younger than 30 years old. Two of them are trainee teachers, one is teacher training student, and one participant was categorized otherwise. One participant teaches in lower secondary schools while the three others teach or will teach in upper secondary schools. The sessions required between 20 and 40 minutes to complete.

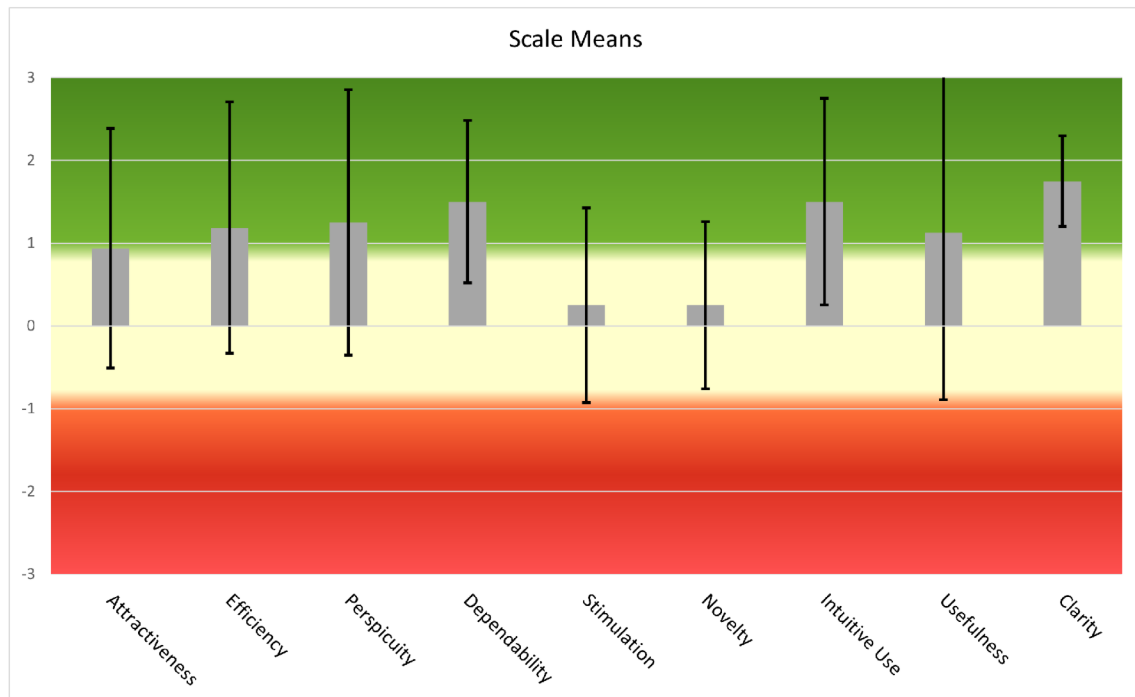
Most participants are creating their learning materials by themselves. They furthermore use materials from online platforms, their colleagues or teaching books. Most of them do not use platforms, where they have to pay for materials. For digital teaching materials, participants use platforms such as Microsoft Teams [28], Schulmanager [42] or Apple's AirDrop [3]. Participants are using digital and interactive applications in general a few times per month, some of them even weekly or daily. When they use digital and interactive applications, they sometimes use WebWriter Explorables or other types of digital worksheets, but also digital games such as Kahoot [23] or Mentimeter [46].

For a statistical evaluation of this study, the collected data was not split by any of the demographic characteristics. With only four participants, the study may not provide any quantitative data but it can provide qualitative information. For finding the most majority problems in usability, a low number of participants is almost as good as a larger number. Using a larger number of participants may provide more insights and they find more (usability) problems, but it also produces higher costs (e.g. time effort). [Bas09] [20]

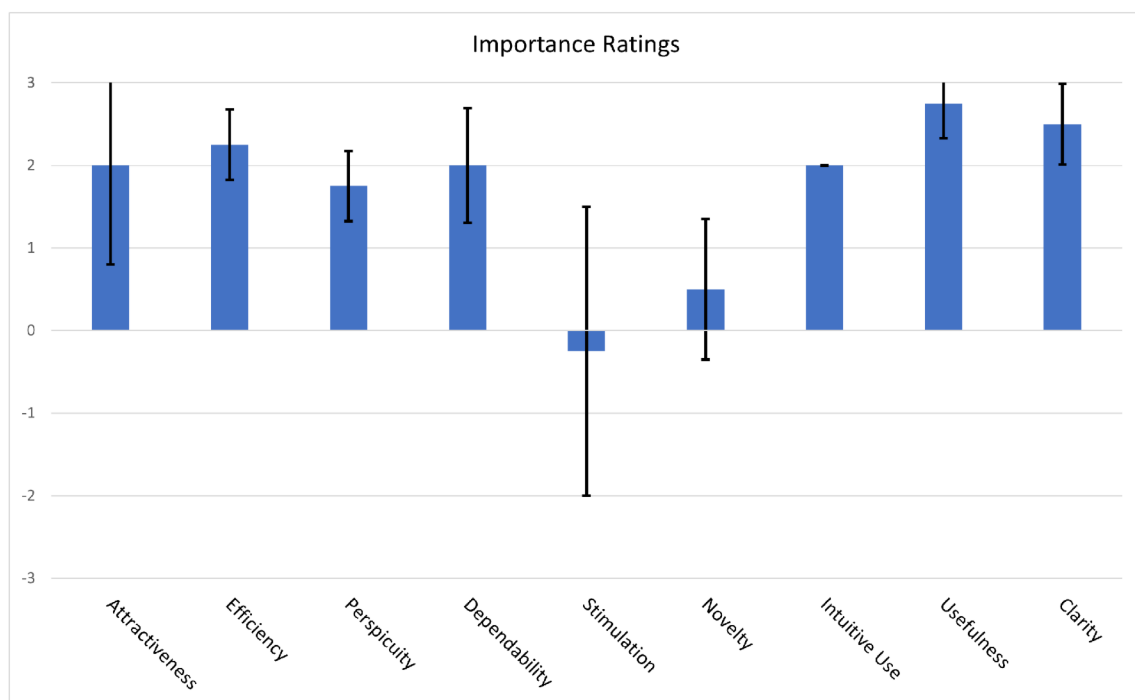
For evaluating the UEQ+, the official website provides an Excel-sheet. This can be downloaded and adjusted. The categories being used can be selected from a list as well as the results can be added. The sheet itself calculates the corresponding results and also provides basic visualizations (see Figure 4.2) for them. The results are shifted on a scale between -3 and 3, where larger numbers are more positive than smaller numbers. Results on the scale around 0 (-0.8 to 0.8) are neutral.

With exception of two scales (stimulation and novelty having a mean score of 0.25) all other scales are having a mean value of 0.92 (attractiveness) or higher (to remember: all values higher than 0.8 are interpreted as positive). The highest mean score has clarity (1.75), directly followed by the categories intuitive use (1.5) and dependability (1.5). The

4.5. Results of the Main Study



(a) Means of the scales (mean over all items in each category)



(b) Importance ratings of each category

Figure 4.2.: Scale Means and Importance Ratings of the selected UEQ+ categories for the prototype evaluation

confidence interval of each scale is presented using the thin black bars. The confidence interval is a statistical measurement, which describes the range of expected values for a measurement under the assumption of a given confidence interval. Here, a confidence interval of 95% is used. Smaller confidence intervals therefore mean, that participants more likely agreed with each other on the rating. [48]

For participants, the most important categories are usefulness (2.75), clarity (2.5), efficiency (2.25), intuitive use (2.0), and attractiveness (2.0). The categories with the lowest importance are stimulation (-0.25) and novelty (0.5). While the most of the categories have a small confidence interval of less than equal 1, attractiveness (confidence of 1.2) and stimulation (confidence of 1.75) are the categories, where participants did not seem to agree with each other in regards of the importance for that scale.

4.6. Interpretation and Discussion

All in all, the results of the UEQ+ can be interpreted as positive, although most of the scale means have relatively large confidence intervals. Only clarity and dependability seem to be those categories, where all participants had a common perception. The larger confidence intervals may be explained by taking a look on the individual results of the study. It seems that one participant evaluated the software as a finished product rather than a prototype. Perhaps this was not made clear in the study.

Stimulation as well as novelty are the only scales that did not score a positive but a neutral result. However, these aspects were also rated as less important by participants. Novelty describes, that a software should have a modern design catching the user's attention. For the design itself, the default implementation of the underlying component library was used. With adjustments, the design can be improved. However, the rating of the importance shows, that the design is not as important as a clear and intuitive user interface.

More important for users are clear and intuitive interfaces and actions as well as the feeling, that the software is beneficial for users without the need of additional effort. I think, the current implementation can fulfill these conditions and the results of the UEQ+ support my assessment.

Additionally to the statistical evaluation using the UEQ+, participants provided qualitative feedback on weaknesses and strengths of the platform. Three of four participants said, that they could imagine to use the exchange platform in their school days. Participants think, that the platform - especially with a larger community - can be an inspiration for learning materials and that the platform is an upgrade for WebWriter, that is currently missing. Participants liked the possibility to create and organize learning groups and provide content using them. Nevertheless, there was also negative feedback and ideas for further improvements from participants: Currently, Explorables must be downloaded such that a user can view them. It may be helpful, to allow opening Explorables without the need for downloading. Furthermore, the search functionality is currently not very advanced. The search could be extended with filters, for example school type, class or subject. One participant missed the possibility to upload their own documents for group content. Furthermore, some functionalities are not directly visible or counter-intuitive

(e.g. toggling the visibility of group content).

In conclusion, the prototype can be considered successfully implemented and positively evaluated. Even though there is still potential for improvement, some functions need to be revised, and some functions have not yet been implemented, the overall results indicate a positive and solid foundation for further development. It should be noted, that the performed study only covered the primary target group: teachers. Other target groups (e.g. students) have not yet been evaluated. Their perspective on usability may differ and should be tested in the future.

Chapter 5 Conclusion

In this final chapter, the thesis is summarized and evaluated with regards to the goal of the thesis. The research question of this master thesis was: "How can the remix workflow be supported by an exchange platform for Explorables?". To answer the question, the thesis required the implementation of a software application as well as its evaluation.

Before starting the actual implementation of the application, a concept was created. The concept includes an analysis of requirements. It defines, who is supposed to use the software application (stakeholder analysis), which functionalities shall be implemented (functional requirements) as well as additional constraints to follow during the implementation (non-functional requirements). It furthermore consists of the development of a first prototype design.

With the underlying concept, the application was implemented. With the existing PocketBase instance as database and backend API as well as Nuxt for the frontend, the concept was realized. The existing database model was extended and the planned functionalities have been implemented. At the end, the exchange platform was evaluated using the User Experience Questionnaire Plus (UEQ+).

Although the evaluation was performed with only a few participants and all of them in a single target group, it indicates that the remix workflow can be supported by providing public access to Explorables as well as integrating learning group management. With access to other user's Explorables, users can get inspiration for creating new learning materials.

During the implementation and evaluation, some problems were already identified and possible solutions were provided. Future development includes solving the existing problems as well as integrating new functionalities.

5.1. Future Work

The implementation of this thesis only contains a little set of possible functionalities. The implemented user interface and functionalities may require some rework and new functionalities could be added in the future. The following list contains possible improvements as well as new features:

- The current implementation of the role-based access control system in groups should be revised. It may lead to inconsistent states as well as unintended behaviour and is missing some major functionalities (e.g. removing a user from a group). For further details, see Chapter 3.4.4.

- Adjust the user interface with regard to usability (e.g. toggling visibility of group contents), design, or internationalization (add translations for texts and provide these in multiple languages).
- Add a more advanced search dialog or page using which Explorables can be filtered (e.g. by subject, class, author, license).
- Add quality-of-life updates:
 - Hinder a user from leaving the current page, when the current page has unsaved changes.
 - Automatically focus input fields (especially in dialogs), when they become visible by a user's action.
- Add better integration of features of the platform. The exchange platform itself, collections and group contents are three components, where Explorables are used. Allow, for example, to add Explorables from inside the exchange platform as group content or allow the selection of Explorables from user's collections in the learning application, when configuring group content.
- Add automatic remix functionality. The user first selects multiple sources (Explorables, images, ...) and then starts a remix action, where these sources could be combined to a new Explorable.
- Add communication possibilities to learning groups, such as news or forums. Teachers can thereby send information to learners and learners can communicate with each other.
- Add a submission system in groups, such that students can hand in their solutions for the corresponding content. Maybe its possible, to implement automatic correction of student's solutions.
- Implement a testing infrastructure for the code and add building the Docker image (Continuous Integration) as well as automatically deploying the built image (Continuous Deployment).
- Currently, only users that are already registered within the platform can be assigned as group members. Using the join functionality, users can create their accounts and join a group afterwards. However, it may be also beneficial to allow group administrators to add users to a group, which are currently not registered. They are then automatically assigned to the group, when they create their account. This may be helpful especially in scenarios, where, for example, a school creates a group for each course and wants to add students as members to their corresponding courses.

The current prototype provides a foundation for future development of the exchange and learning platform for WebWriter Explorables. It integrates into the existing ecosystem of WebWriter, allows the sharing and searching of Explorables and includes a built-in learning management system. The used technologies provide room for improving existing and implementing additional features towards a feature-rich application, designed for educational workflows.

Appendix A Bibliography

- [All23] Joshua Allius. Development of a server component for the WebWriter system. Bachelorarbeit, RWTH Aachen University, Aachen, 2023. Veröffentlicht auf dem Publikationsserver der RWTH Aachen University; Bachelorarbeit, RWTH Aachen University, 2023.
- [Bas09] J. Bastien. Usability testing: A review of some methodological and technical aspects of the method. International journal of medical informatics, 79:e18–23, 05 2009.
- [Bro95] John Brooke. SUS: A quick and dirty usability scale. Usability Eval. Ind., 189, 11 1995.
- [Cla95] Max B. E. Clarkson. A Stakeholder Framework for Analyzing and Evaluating Corporate Social Performance. The Academy of Management Review, 20(1):92–117, 1995. Publisher: Academy of Management.
- [fOKF23] Fraunhofer-Institut für Offene Kommunikationssysteme FOKUS. LOGINEO NRW - Zukunftsscheck, 2023.
- [GRB21] Linda Grogorick and Susanne Robra-Bissantz. Digitales lernen und lehren: Führt corona zu einer zeitgemäßen bildung? HMD Praxis der Wirtschaftsinformatik, 58(6):1296–1312, Dec 2021.
- [Hab24] Martin Habersaat. Itslearning 2024 - Nutzerzahlen stark rückläufig. <https://www.martinhabersaat.de/2024/11/04/itslearning-2024-nutzerzahlen-stark-ruecklaeufig/>, Nov 2024. accessed: 2025-10-06.
- [Ham11] Carol Hammer. The importance of participant demographics. American journal of speech-language pathology / American Speech-Language-Hearing Association, 20:261, 11 2011.
- [iee18] ISO/IEC/IEEE International Standard - Systems and software engineering – Life cycle processes – Requirements engineering. ISO/IEC/IEEE 29148:2018(E), pages 1–104, 2018.
- [JKT21] Michael Eichhorn Ralph Müller Jan Konrad, Angela Rizzo and Alexander Tillmann. Digitale Technologien und Schule. Ein Schulentwicklungsprozess aus der Perspektive der Akteur-Netzwerk-Theorie. 2021.

- [Jü23] Volker Jürgens. Warum Passwort-Organisation im Schulalltag herausfordernd sein kann – Kollegien ächzen. <https://www.news4teachers.de/2023/04/warum-passwort-organisation-im-schulalltag-herausfordernd-sein-kann-kollegien-aechzen/>, April 2023. accessed: 2025-10-06.
- [LHS08] Bettina Laugwitz, Theo Held, and Martin Schrepp. Construction and Evaluation of a User Experience Questionnaire. volume 5298, pages 63–76, 11 2008.
- [MB02] Martin Maguire and Nigel Bevan. User Requirements Analysis, pages 133–148. Springer US, Boston, MA, 2002.
- [Sch23] Martin Schrepp. User Experience Questionnaire Handbook, 09 2023.
- [SDW⁺10] Alexander Schatten, Markus Demolsky, Dietmar Winkler, Stefan Biffl, Erik Gostischa-Franta, and Thomas Östreicher. Software-Architektur, pages 199–227. Spektrum Akademischer Verlag, Heidelberg, 2010.
- [SRS23] Frederic Salmen, René Roepke, and Ulrik Schroeder. Webwriter: A system to author and remix explorables – requirements & first prototype. In Workshops der 21. Fachtagung Bildungstechnologien (DELFI), pages 39–48. Gesellschaft für Informatik e.V., Bonn, 2023.
- [SRS24] Frederic Salmen, Rene Roepke, and Ulrik Schroeder. Webwriter: Authoring and remixing explorables. In Rafael Ferreira Mello, Nikol Rummel, Ioana Jivet, Gerti Pishtari, and José A. Ruipérez Valiente, editors, Technology Enhanced Learning for Inclusive and Equitable Quality Education, pages 247–253, Cham, 2024. Springer Nature Switzerland.
- [ST23] Martin Schrepp and Jörg Thomaschewski. Handbook for the modular extension of the User Experience Questionnaire, 12 2023.
- [Tro23] Marc Troll. User-centric enhancement of the Coach-VR. Bachelorarbeit, RWTH Aachen University, Aachen, 2023. Veröffentlicht auf dem Publikationsserver der RWTH Aachen University; Bachelorarbeit, RWTH Aachen University, 2023.
- [WHI18] David Wiley and John Levi Hilton III. Defining oer-enabled pedagogy. The International Review of Research in Open and Distributed Learning, 19(4), Sep. 2018.

Appendix B Online References

- [1] 7 Gestalt Principles. <https://www.usertesting.com/blog/gestalt-principles>. accessed: 2/11/2025.
- [2] About GitHub and Git. <https://docs.github.com/en/get-started/start-your-journey/about-github-and-git>. accessed: 5/11/2025.
- [3] AirDrop. <https://support.apple.com/de-de/guide/iphone/iphcd8b9f0af/ios>. accessed: 3/11/2025.
- [4] Amazon. <https://amazon.de/>. accessed: 19/09/2025.
- [5] Amazon Statistics. <https://amzscout.net/blog/amazon-statistics/>. accessed: 19/09/2025.
- [6] Angular vs. React vs. Vue.js – Choosing a JavaScript Framework for Your Project. <https://relevant.software/blog/angular-vs-react-vs-vue-js-choosing-a-javascript-framework-for-your-project/>. accessed: 15/10/2025.
- [7] Bundesministerium für Digitales und Staatsmodernisierung. <https://bmds.bund.de>. accessed: 12/05/2025.
- [8] Client.ts - pocketbase/js-sdk - Github. <https://github.com/pocketbase/js-sdk/blob/master/src/Client.ts#L505>. accessed: 23/10/2025.
- [9] Composables | Vue.js. <https://vuejs.org/guide/reusability/composables.html>. accessed: 23/10/2025.
- [10] Composition API: setup() | Vue.js. <https://vuejs.org/api/composition-api-setup>. accessed: 24/10/2025.
- [11] Digitalisierungsschub durch Corona? Eine wissenschaftliche Perspektive. <https://www.empirische-bildungsforschung-bmbfsfj.de/de/Digitalisierungsschub-durch-Corona-Eine-wissenschaftliche-Perspektive-2060.html>. accessed: 6/11/2025.
- [12] Digitalpakt Schule. <https://digitalpaktschule.de/>. accessed: 12/05/2025.
- [13] Docker. <https://www.docker.com/>. accessed: 21/10/2025.
- [14] Docker Workshop. <https://docs.docker.com/get-started/workshop/>. accessed: 21/10/2025.
- [15] eduki. <https://eduki.com/de>. accessed: 11/05/2025.

- [16] Explorables. <https://webwriter.app/docs/explorables/explorables/>. accessed: 23/04/2025.
- [17] Extend with JavaScript - Overview - Docs - PocketBase. <https://pocketbase.io/docs/js-overview/>. accessed: 21/10/2025.
- [18] Figma. <https://www.figma.com/de-de/>. accessed: 07/10/2025.
- [19] Going to Production - Docs - PocketBase. <https://pocketbase.io/docs/going-to-production/#using-docker>. accessed: 21/10/2025.
- [20] How Many Test Users in a Usability Study? <https://www.nngroup.com/articles/how-many-test-users/>. accessed: 3/11/2025.
- [21] Introduction - Authentication - User Impersonation - PocketBase. <https://pocketbase.io/docs/authentication/#users-impersonation>. accessed: 23/10/2025.
- [22] JS SSR - issues and recommendations when interacting with PocketBase - GitHub. <https://github.com/pocketbase/pocketbase/discussions/5313>. accessed: 23/10/2025.
- [23] Kahoot! | Learning games | Make learning awesome! <https://kahoot.com/de/>. accessed: 3/11/2025.
- [24] Lit. <https://lit.dev/>. accessed: 15/10/2025.
- [25] LocalAuthStore.ts - pocketbase/js-sdk - Github. <https://github.com/pocketbase/js-sdk/blob/master/src/stores/LocalAuthStore.ts>. accessed: 23/10/2025.
- [26] LOGINEO NRW LMS. <https://www.logineo.schulministerium.nrw.de/LOGINEO-NRW/NEU-LOGINEO-NRW-LMS-Lernmanagementsystem/>. accessed: 12/05/2025.
- [27] Microservices vs. monolithic architecture. <https://www.atlassian.com/microservices/microservices-architecture/microservices-vs-monolith>. accessed: 06/10/2025.
- [28] Microsoft Teams. <https://www.microsoft.com/de-de/microsoft-teams/group-chat-software>. accessed: 3/11/2025.
- [29] Moodle. <https://moodle.org/>. accessed: 11/05/2025.
- [30] Number of monthly unique YouTube users. <https://www.statista.com/statistics/280685/number-of-monthly-unique-youtube-users/>. accessed: 19/09/2025.
- [31] Nuxt. <https://nuxt.com/>. accessed: 16/10/2025.
- [32] Nuxt and hydration - Best Practices v3. <https://nuxt.com/docs/3.x/guide/best-practices/hydration>. accessed: 23/10/2025.
- [33] nuxt-i18n-micro - Nuxt modules. <https://nuxt.com/modules/nuxt-i18n-micro>. accessed: 22/10/2025.

- [34] Nuxt instance is unavailable after await in nuxt plugin - GitHub. <https://github.com/nuxt/nuxt/issues/30796>. accessed: 24/10/2025.
- [35] pages - Nuxt Directory Structure v3. <https://nuxt.com/docs/3.x/guide/directory-structure/pages>. accessed: 22/10/2025.
- [36] Passwordless Login. <https://auth0.com/docs/authenticate/passwordless>. accessed: 06/10/2025.
- [37] Pocketbase. <https://pocketbase.io/>. accessed: 17/10/2025.
- [38] PrimeVue. <https://primevue.org/>. accessed: 22/10/2025.
- [39] React. <https://react.dev/>. accessed: 15/10/2025.
- [40] REST. <https://restfulapi.net/>. accessed: 16/09/2025.
- [41] Schule Nordrhein-Westfalen - Statistiken. <https://statistik.nrw/schulen-nordrhein-westfalen>. accessed: 12/05/2025.
- [42] Schulmanager Online. <https://www.schulmanager-online.de/>. accessed: 3/11/2025.
- [43] Shoelace. <https://shoelace.style/>. accessed: 22/10/2025.
- [44] SSR Integration - Readme pocketbase/js-sdk - GitHub. <https://github.com/pocketbase/js-sdk#ssr-integration>. accessed: 23/10/2025.
- [45] State Management - Get started with Nuxt v3. <https://nuxt.com/docs/3.x/getting-started/state-management>. accessed: 23/10/2025.
- [46] Tool für interaktive Präsentationen Publikumsbeteiligung - Mentimeter . <https://www.mentimeter.com/de-DE>. accessed: 3/11/2025.
- [47] UEQ+. <https://ueqplus.ueq-research.org/>. accessed: 29/10/2025.
- [48] Understanding Confidence Intervals | Easy Examples Formulas. <https://www.scribbr.com/statistics/confidence-interval/>. accessed: 7/11/2025.
- [49] Unterschied zwischen einem Framework und einer Library. <https://www.norules-webdesign.de/blog/unterschied-zwischen-einem-framework-und-einer-library>. accessed: 15/10/2025.
- [50] Usability - digital.gov. <https://digital.gov/topics/usability/>. accessed: 28/10/2025.
- [51] Usability - digital.gov. <https://fokus-ux.de/usability-user-experience>. accessed: 28/10/2025.
- [52] User Experience Questionnaire (UEQ). <https://www.ueq-online.org/>. accessed: 29/10/2025.
- [53] Vue.js Documentation - Server Side Rendering. <https://vuejs.org/guide/scaling-up/ssr.html>. accessed: 16/10/2025.
- [54] Was ist der UEQ? <https://easy-feedback.de/umfrage-beispiele/ueq-fragebogen-vorlage/ueq-definition/>. accessed: 29/10/2025.

- [55] Was ist der UEQ? <https://www.nngroup.com/articles/pilot-testing/>. accessed: 29/10/2025.
- [56] Web Components. https://developer.mozilla.org/en-US/docs/Web/API/Web_components. accessed: 23/04/2025.
- [57] WebWriter - Documentation - Widgets. <https://webwriter.app/docs/widgets/widgets/>. accessed: 23/04/2025.
- [58] WebWriter - Packages. <https://webwriter.app/packages/>. accessed: 06/11/2025.
- [59] Wie funktionieren Google Crawler. <https://www.hanseranking.de/wie-funktionieren-google-crawler/>. accessed: 16/10/2025.
- [60] YouTube. <https://youtube.com/>. accessed: 19/09/2025.
- [61] Zoom. <https://www.zoom.com/>. accessed: 2/11/2025.

Appendix C Listings

```
1 import { BaseAuthStore } from 'pocketbase';
  import type { IncomingMessage, ServerResponse } from 'node:http';

  export class CookieAuthStore extends BaseAuthStore {
5     constructor(req?: IncomingMessage, res?: ServerResponse) {
        super();

        if (import.meta.server && req) {
            const cookieHeader = req.headers.cookie || '';
10         try {
            super.loadFromCookie(cookieHeader);
        } catch (err) {
            console.warn('Could not load auth from cookie:', err);
        }
15     }

    if (import.meta.client) {
        const cookie = useCookie('pb_auth');
        if (cookie.value) {
20         try {
            // cookie.value can be an object
            const cookieString =
                typeof cookie.value === 'string'
                    ? cookie.value
25                 : encodeURIComponent(JSON.stringify(cookie.value));
            super.loadFromCookie(`pb_auth=${cookieString}`);
        } catch (err) {
            console.warn('Client load failed:', err);
        }
30     }
  }

  this.onChange(async () => {
    // create pocket base auth cookie as
35    const cookieStr = this.exportToCookie({
      httpOnly: false,
      path: '/',
      sameSite: 'strict',
    });
40

    // write cookie on server (is this possible in general?)
    if (import.meta.server && res) {
```

```

        res.setHeader('Set-Cookie', cookieStr);
    }
45
    // write cookie on client
    if (import.meta.client) {
        const cookieValue = decodeURIComponent(
            cookieStr.split('=')[1]?.split(';')[0] || '',
50
        );
        const cookie = useCookie('pb_auth', {
            path: '/',
            sameSite: 'strict',
            httpOnly: false,
55
        });
        cookie.value = cookieValue;
    }
    });
}
60 }

```

Listing C.1: Implementation of the `CookieAuthStore`

```

1 import PocketBase from 'pocketbase';
import { CookieAuthStore } from '~/composables/pocketbase/
  CookieAuthStore';

export const usePocketbaseClient = () => {
5   if (import.meta.server) {
      const nuxtApp = tryUseNuxtApp();
      const req = nuxtApp?.ssrContext?.event?.node.req;
      const res = nuxtApp?.ssrContext?.event?.node.res;

10      const pocketbase = new PocketBase(
          useRuntimeConfig().apiServer,
          new CookieAuthStore(req, res),
      );
      pocketbase.autoCancellation(false);
15      return pocketbase;
    }

    if (!(globalThis as any)._pb) {
      (globalThis as any)._pb = new PocketBase(
20        useRuntimeConfig().public.apiServer,
        new CookieAuthStore(undefined, undefined),
      );
      (globalThis as any)._pb.autoCancellation(false);
    }

25    return (globalThis as any)._pb as PocketBase;
  };

```

Listing C.2: Implementation of the Composable

Appendix D Digital Appendix

The following files are included in the digital appendix. The digital appendix is uploaded as a ZIP-file (digital.zip).

- **source.zip**: This ZIP-file includes the source code of the project at the moment of the official hand-in of this thesis. It is downloaded from GitHub.
- **MA-Marc-Troll.pdf**: The final version of the thesis, that was handed in.
- **survey-results.xlsx**: The raw results of the survey. This file is secured with a password due to privacy. The password is given to the Lehr- und Forschungsgebiet Lerntechnologien.
- **survey.lss**: The export from limesurvey, that includes the survey structure.

Appendix E Assistive Tools

During the implementation, tools were used that were not worth mentioning in the thesis writing itself. The following provides a list of those tools.

- **IntelliJ IDEA:** IntelliJ was used to implement the source code of the platform.
- **GitHub:** The version control system GitHub was used to store and version the source code.
- **Overleaf:** Overleaf was used for writing the thesis in LaTeX.
- **ChatGPT:** The large language model was used for discussions during the implementation as well as the writing of the thesis. It was used to provide impulses for thought. Discussions included bug fixes during the implementation as well as discussing formulations and the structure of the thesis. ChatGPT was **not** used for actually writing the thesis.
- **Zotero:** Zotero was used for organizing and storing bibliography, that was used within this thesis.

Eidesstattliche Versicherung

Troll, Marc

394593

Name, Vorname

Matrikelnummer (freiwillige Angabe)

Ich versichere hiermit an Eides Statt, dass ich die vorliegende ~~Arbeit/Bachelorarbeit/~~ Masterarbeit* mit dem Titel

Development of an Exchange Platform for WebWriter-Explorables

selbstständig und ohne unzulässige fremde Hilfe (insbes. akademisches Ghostwriting) erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt. Für den Fall, dass die Arbeit zusätzlich auf einem Datenträger eingereicht wird, erkläre ich, dass die schriftliche und die elektronische Form vollständig übereinstimmen. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Aachen , November 11, 2025

Ort, Datum

Marc Troll

Unterschrift

*Nichtzutreffendes bitte streichen

Belehrung:

§ 156 StGB: Falsche Versicherung an Eides Statt

Wer vor einer zur Abnahme einer Versicherung an Eides Statt zuständigen Behörde eine solche Versicherung falsch abgibt oder unter Berufung auf eine solche Versicherung falsch aussagt, wird mit Freiheitsstrafe bis zu drei Jahren oder mit Geldstrafe bestraft.

§ 161 StGB: Fahrlässiger Falscheid; fahrlässige falsche Versicherung an Eides Statt

(1) Wenn eine der in den §§ 154 bis 156 bezeichneten Handlungen aus Fahrlässigkeit begangen worden ist, so tritt Freiheitsstrafe bis zu einem Jahr oder Geldstrafe ein.

(2) Strafflosigkeit tritt ein, wenn der Täter die falsche Angabe rechtzeitig berichtigt. Die Vorschriften des § 158 Abs. 2 und 3 gelten entsprechend.

Die vorstehende Belehrung habe ich zur Kenntnis genommen:

Aachen , November 11, 2025

Ort, Datum

Marc Troll

Unterschrift