Enhancing Digital Learning: A User Management and Access System for Remote Laboratories

Marek Achilles, Emilia Weinhold, Kathleen Hallmann, Debora Beger, Chantal König, Florian Matthes, Celina Scholz, Max Schweizer, Rico Beier-Grunwald, Alexander Lampe, Marc Ritter, Matthias Vodel, Christian Roschke
University of Applied Sciences Mittweida
Mittweida, Germany
{machille, eweinhol, kkeilber, dbeger, ckoenig, fmatthe2, cscholz6, mschwei1, beier1, lampe, ritter, vodel, roschke}@hs-mittweida.de

***Abstract*—The digitization of teaching and education is continuously increasing and numerous management systems for digital courses are available. However, the limits of these systems are quickly reached or the systems are too expensive when remote interaction with real laboratory hardware is required. We present a web-based Remote Lab System (RLS) created with open-source tools, which can be integrated flexibly into the existing university infrastructure and realizes the administration, planning and control of the online access to laboratory devices.**

***Index Terms*—Distance learning, user management system, remote/online hardware access, web application**

I. Introduction

Remote web-based teaching provides flexibility and can be facilitated by open source or commercial learning management systems (LMS) and tools like Moodle, Adobe Captivate, MS Teams etc. for the majority of lecture formats. However, for engineering students, laboratory work with physical devices is an essential part of their education that must also be available online. While some laboratory hardware can be accessed via web user interfaces, it is often difficult to integrate them into standard LMSs and ensure that only authorized users operate the devices. Commercial solutions are available from the device manufacturers [1], but can be expensive and difficult to integrate into existing university IT infrastructure. To address these challenges, we developed an open-source web-based RLS with key LMS functions and the flexibility to later integrate with web interfaces of different devices. Our system was designed to be cost-effective and to leverage existing university IT infrastructure to maximize efficiency.

In the following sections, we provide a detailed explanation of the architecture and software components used and conclude with examples of potential applications and extension points to demonstrate the versatility of our approach.

II. System Architecture and Data Model

The RLS software architecture is presented in Fig. 1. The management module includes a frontend for user interaction, which communicates with the backend through an API that encapsulates business logic and database communication. The hardware accessibility module consists of proxy servers for both HTTP and WebSocket requests that provide access to the lab hardware. A reverse proxy server is used to add security features such as connection encryption and authorization via the university's single sign-on system Shibboleth. Choosing an appropriate network configuration is crucial for preventing users from bypassing the system through direct access.

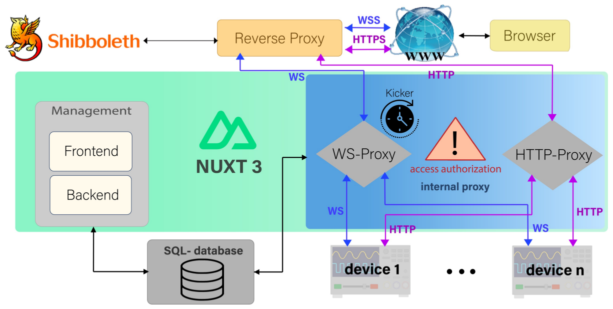

Fig. 1. System architecture of the RLS.

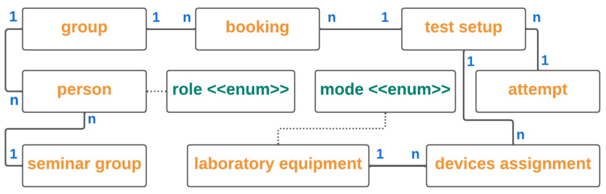

Fig. 2. Data model of user management subsystem.

The data model of the user management subsystem is depicted in Fig. 2. The system is designed for groups of people with assigned roles of either student or lecturer. Student groups have the ability to reserve timeslots for experimental setups. Each setup consists of devices assigned by the lecturer based on the specific experiment to be conducted. The number of available physical devices determines how many setups can be booked in parallel. Lab devices can be assigned to multiple experiments, as long as these experiments cannot be performed simultaneously.

III. Implementation

Various frameworks were utilized for the implementation of the RLS to expedite development and enhance reliability,

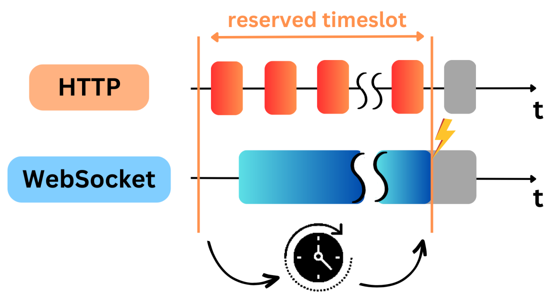

Fig. 3. Hardware access handling: the system verifies permission at the start of each connection, since HTTP connections are short-lived, they can only slightly exceed the time limit; in contrast, WebSocket connections are long-lived and require active termination to comply with the time limit.

allowing us to concentrate on addressing the core issues rather than common problems. To achieve a consistent design and seamless functionality across the user interface, the frontend was build using the Javascript framework Vue in conjunction with Primevue, a library of modern and customizable UI components. The full-stack web application framework Nuxt was utilized to implement the backend and enable communication between the frontend and backend components through an API, which facilitates smooth integration and efficient communication. Special attention was given to secure data manipulation and explicit error handling. To streamline communication with the database, we utilized the schema-driven Object-Relational-Mapping library, Prisma. This library provides a fully-typed object-based abstraction over a relevant subset of SQL, enabling developers to work with the database in a more intuitive and efficient manner. Authentication and authorization are handled by the modular reverse proxy server, which, in our case, utilizes a plugin for the Shibboleth system. This design provides a flexible and modular approach, allowing for easy integration with other authentication systems with minimal impact on the application code. The system will therefore accommodate the varying authorization procedures across different universities.

The most challenging aspect of the development process was implementing redirection and access control of the existing hardware user interfaces. This required careful inspection and a deep understanding of the devices' software architecture and functionality. The development process required the implementation of two separate proxy server components. The first component was an HTTP proxy server responsible for access control and redirecting requests to the appropriate devices' start pages based on internal routing data. Achieving seamless integration of the system required intercepting responses and adjusting link paths accordingly. To enable interactive monitoring and control of the lab hardware through the device user interface, a long-lived WebSocket connection is established. As a result, a WebSocket proxy had to be implemented as the second component to forward real-time communication between the devices and the user. Due to the long-lived nature of WebSocket connections, it was necessary to give special attention to terminating open socket connections once they reached the reserved time limit (see Fig. 3).

IV. Quality Assessment and Evaluation

To ensure the integrity and reliability of our API, each individual endpoint was subjected to extensive automated testing using a variety of test cases, including positive, negative, and edge cases. To facilitate fine-grained control of test cases, programming standards were established that require distinguishable error messages. By adhering to these standards, test cases can differentiate between different error causes, enabling more thorough and detailed assessment. The testing process was automated using the open-source tool Vitest with a separate database in a containerized environment to help avoiding any potential conflicts with the production database.

To evaluate the usability of the software, various user tests were conducted including a prototype evaluation with 45 students who tried out functions such as group registration and appointment booking, as ll as an evaluation of experiment configuration by 3 lecturers. To gather feedback and identify areas for improvement, the open-source tool AttrakDiff was used in conjunction with individual questionnaires. The derived portfolio representation of this evaluation is shown in Fig. 4 indicating that the developed application was considered to possess a high hedonic as well as pragmatic quality.

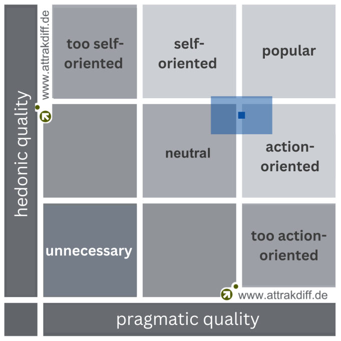

Fig. 4. Portfolio representation of hedonic (perceived stimulation and user identification) and pragmatic quality (usability and goal achievement).

V. Summary

Our system supports laboratory work for students, regardless of location. Users can register and access assigned devices during booked time slots. Instructors can manage devices, users, bookings, and experimental scheduling. The software has already been implemented for remote control of function generators and oscilloscopes, and can be easily adapted for other lab equipment or standard embedded systems with a web user interface, such as Raspberry Pi. Future extensions to the software may include a chat function and document management, providing additional value and enhancing user experience.

References

[1] KEYSIGHT Technologies, "PathWave Lab Operations für Remote Learning, PW9112EDU"