



# Development of a Learning Management System within a Sales Enablement Platform

Markéta Sovová

Bachelor's thesis

June 2024

Information and Communications Technology

**Sovová, Markéta**

### **Development of a Learning Management System within a Sales Enablement Platform**

Jyväskylä: Jamk University of Applied Sciences, June 2024, 44 pages

Degree Programme in Information and Communications Technology. Bachelor's thesis.

Permission for open access publication: Yes

Language of publication: English

#### **Abstract**

The development of a Learning Management System (LMS) integrated into a Sales Enablement Platform was undertaken to enhance the capabilities of the SalesCue platform, a Finnish sales enablement tool. The project aimed to create a web-based LMS that would provide structured educational content to improve users' skills and knowledge. The motivation for this development was driven by professional interests and the growing importance of e-learning solutions. The implementation involved using modern technologies such as MongoDB for backend data management, tRPC for type-safe API calls, and React Native for frontend development. The project focused on building core functionalities like course creation, lesson progression, and quiz assessments. The LMS was designed to be integrated seamlessly into the existing SalesCue ecosystem, maintaining a consistent user experience. Results showed successful integration of essential LMS features, although some functionalities like quiz completion and settings options were still under development. The project demonstrated the usefulness and potential benefits of incorporating an LMS into a sales enablement platform, enhancing its value and utility for users. Conclusively, the development process highlighted the challenges and learning experiences in full-stack development, underscoring the importance of iterative testing and feedback. Future development phases are expected to expand the LMS's capabilities, further enriching the SalesCue platform's offerings.

#### **Keywords/tags (subjects)**

Learning Management System, Sales Enablement Platform, MongoDB, tRPC, React Native, e-learning, web development

## Contents

<b>1</b>	<b>Introduction .....</b>	<b>5</b>
<b>2</b>	<b>Theoretical framework .....</b>	<b>8</b>
2.1	Overview of Learning Management System .....	8
2.2	Why does a Sales Enablement Platform need an LMS? .....	12
2.3	Technological stack overview.....	17
2.4	UX and interface design .....	20
<b>3</b>	<b>Implementation.....</b>	<b>24</b>
3.1	Database design .....	25
3.2	Backend development .....	29
3.3	Frontend development .....	35
<b>4</b>	<b>Conclusion.....</b>	<b>41</b>
<b>5</b>	<b>References.....</b>	<b>43</b>

## Figures

Figure 1:	LMS project's processes and phases.....	15
Figure 2:	Fetching response from the API call automatically offers its type structure. ....	18
Figure 3:	Course Management View in Figma .....	21
Figure 4:	Lesson editing View in Figma .....	21
Figure 5:	Course completion view in Figma.....	22
Figure 6:	LMS Architectural design .....	24
Figure 7:	LMS Database scheme. ....	26
Figure 8:	tRPC initialization. ....	30
Figure 9:	tRPC API router. ....	31
Figure 10:	Course router, list method's input.....	33
Figure 11:	List method, the formation of query. ....	34
Figure 12:	List method, return statement. ....	34
Figure 13:	tRPC client initialization. ....	35
Figure 14:	Custom React hook useCourses.....	36
Figure 15:	Action method to update a course. ....	36
Figure 16:	Course row.....	38

**Tables**

Table 1: Course, lesson, and category router endpoints.....32

# 1 Introduction

This thesis focuses on the development of a web-based Learning Management System (LMS) integrated into a Sales Enablement Platform. Sales enablement is a continuous process of providing sales teams the resources to enhance sales, marketing, and deal closures. The sales enablement applications typically offer tools for managing sales content and for presenting, sharing, and analyzing materials. The central concept of the LMS is to allow administrators to create structured educational content, which includes courses, lessons, and quizzes. The purpose of the LMS is to increase the skills and knowledge of the users. Once created, users can enroll in these courses, proceed through the lessons at their own pace and complete quizzes that assess their understanding of the topic.

The client requesting the project is a Finnish company, that will be called by a pseudonym SalesCue throughout the thesis in order to protect the company's privacy and sensitive data. SalesCue was founded in 2012 and its main product is a Sales Enablement Platform, which will be called SalesCue in the thesis. SalesCue is a cross-platform software available on Web, iOS, Windows, and Android devices, however LMS will be developed only for the Web platform.

The motivation for this project comes from both professional and personal interests. I have been employed in SalesCue for the last four and a half years as a software developer. The development of LMS is seen as a key step to enhance the platform's capabilities and meet our customers' needs more effectively. The addition of an LMS is expected to increase the overall value of the SalesCue platform. The project also reflects the growing importance of e-learning solutions in today's technology industry. On a personal level, the LMS project is significant for me as well. It is my first experience with full-stack development, which builds on my previous work in frontend development.

While Learning Management Systems have been extensively researched from perspectives of usability, design, user satisfaction and learning efficiency, there are not many that would focus on LMS development from a software engineering point of view. Some relevant works that I will review are *Development of Student-centred Language Learning Environment* (Seppälä, 2017) and *Full stack development in TypeScript with tRPC and React Native* (Nivasalo, 2023). The latter utilizes a

very similar technological stack, although in a different application. I have not come across a thesis, whose main focus would be in developing LMS within a sales enablement platform like SalesCue.

In the thesis I will focus on a limited subject of the larger project of integrating a full-featured Learning Management System into the SalesCue platform. Due to the broad scale and complexity of a complete LMS development, I will concentrate on building and demonstrating the core functionalities necessary for a minimal viable product. Specifically, the project will address the creation of courses, including lessons and quizzes, and will enable users to enroll in, progress through, and complete courses. This approach serves as a proof of concept that lays the foundational framework for future expansion. Additional features such as course prerequisites, analytics, certificates, and customizable main pages, while planned for future development, fall outside the scope of this thesis. Moreover, the limited time and space for this project doesn't allow for thorough testing and iteration based on feedback. Other team members also contribute to the development, mostly in regard to architectural and UI design, cloud configuration and user authentication mechanism.

In the technical implementation of the project, I will use several modern technologies to build the integrated Learning Management System within the SalesCue platform. The backend will be supported by MongoDB, a NoSQL database known for its high performance and flexibility in managing large volumes of unstructured data. The communication between the client and server will be managed through tRPC, which performs type-safe API calls, ensuring our data exchanges are reliable and error-free. The frontend will be built using React Native written in TypeScript, which is consistent with the whole SalesCue application. Already existing components will be used, especially in the development of administrative features of the LMS. However, I will also build new components specifically designed for the LMS features, like displaying course content and completion of interactive quizzes. The combination of existing and new components will enable LMS to have the familiar look and feel of the whole SalesCue ecosystem, while also addressing the specific needs of educational content delivery.

Following this introduction, the thesis is structured into several chapters, each focusing on a specific aspect of the project. The first main chapter, *Theoretical Framework*, presents the basic concepts and explores the most important topics. I will delve into the LMS and e-learning on a general level and focus on its key functionalities. After that I will discuss the central point of the thesis, which is the benefits and reasons for implementing LMS within a Sales Enablement Platform. This is followed by a chapter detailing the technologies used in the project, including MongoDB, tRPC, React Native, and TypeScript. The final part of the *Theoretical Framework* examines the design principles and user experience considerations for the LMS. In the subsequent chapter, *Implementation*, I will inspect the practical aspects of the project. I will cover the architecture setup, database models and the specifics of backend and frontend development. The thesis wraps up with the *Conclusion* chapter, which synthesizes the key findings of the research and the project's outcomes. I will evaluate the success of the development, reflect on the learning experience, and discuss potentials for future development.

## 2 Theoretical framework

### 2.1 Overview of Learning Management System

The theoretical part of the thesis begins with an exploration of the Learning Management System in corporate settings, discussing the advantages of e-learning systems with regards to employee training, skill development and performance tracking. I will also examine the standard features found in most LMS (like course and lesson creation, user management and analytics) and how these features support educational and sales objectives.

The rapid changes in the Information and Communication Technology (ICT) field have had a dramatic impact on various sectors, including educational and professional development activities like teaching, training, and learning (Islam, 2012). Organizations continually adapt to these technological changes by integrating modern ICT tools, such as e-learning platforms, video conferences and other digital resources into their development programs. This integration not only increases productivity and efficiency but also modernizes traditional educational and training methods. (Bezhovski & Poorani, 2016)

In recent years, the importance of e-learning has grown significantly, in the wake of events like the COVID-19 pandemic, which create a need for flexible remote solutions. This shift has led to a considerable increase in the user base of virtual learning platforms. (Mitra, 2022.) E-learning's flexibility accommodates a variety of learning needs and styles, making education and training more accessible to a wider audience. One of the key advantages of e-learning is its cost-effectiveness. It dramatically reduces the need for physical resources, and associated travel and accommodation expenses, compared to traditional training methods. Additionally, the scalability of virtual learning allows courses to be brought to many learners simultaneously without corresponding increase in costs. (Islam, 2012)

Another significant advantage of e-learning is its time-efficiency. The process of creating and presenting content is optimized, saving trainers a substantial amount of time. E-learning platforms



enable continuous learning and upskilling, which is accessible anytime and anywhere. Furthermore, the digital content can be easily updated and reused, making sure that all users have access to the same training and latest information, which is crucial for maintaining the company's standards and compliance.

Even though e-learning platforms offer various potential benefits, there also exist numerous reasons why educators might hesitate to adopt them. These include difficulty in use, issues with their reliability, a significant amount of time needed to learn and effectively use new tools, and lack of compatibility with specific teaching and learning goals (Morgan, 2003).

A Learning Management System is an essential tool in the realm of digital education and corporate training. It is a software application designed to handle various aspects of the learning process and user management. While the specific features of LMS can vary depending on their complexity and intended use, most share common functionalities that address the core educational and training needs.

A comprehensive LMS should offer centralized administration, quick assembly and delivery of learning content and thus enable integration of training on web or other platforms. It should also support standards, portability and allow users to create personalized content and the reuse of knowledge. (Ellis, 2009) Some of the most significant and widely used LMSs include an open-source *Moodle*, *Canvas* by Instructure and *Blackboard Learn*.

The importance of usability and a user-friendly interface in developing LMS cannot be overstated. Usability in an LMS refers to how intuitive and easy it is for users to navigate and use the system to achieve their learning objectives. (Islam, 2012) Intuitive design contributes to the efficiency of learners as well as administrators. For learners, this means spending less time understanding the system's operations and more time engaging in actual learning activities. For educators and administrators, it simplifies crucial tasks such as course development, monitoring users' progress, and data reporting.

A well-designed LMS should be accessible to users with varying levels of technical skills and abilities. Moreover, a user-friendly LMS is crucial for adoption and sustained use. Users who find their

interactions with a system satisfying tend to keep using it and engage with the learning materials. (Zanjani, 2017.)

There are several key features that a comprehensive LMS should utilize and which are at least partly going to be developed as a part of this thesis. Some of the essential features Ellis (2009, pp. 2–3) points out are the following:

1. **Integration with HR system**
2. **Administration Tools**
3. **Content Delivery**
4. **Content Development**
5. **Assessment Capabilities**

It's important to keep in mind that the LMS being developed within the SalesCue platform is not a standalone project but an integral component of a broader Sales enablement tool. Unlike traditional, comprehensive LMS tools designed for extensive educational purposes, this LMS emphasizes simplicity and accessibility. Its functionalities are designed to ensure a short learning curve specifically meeting the needs of sales teams. The primary aim is not to compete with established LMS providers but rather to enhance the SalesCue Sales Enablement Platform, making it a more versatile and competitive solution in the market of sales tools.

Integration with **HR systems** in LMS means establishing a seamless connection between the learning platform and organization's user data. This integration is vital, as it enables users to directly engage with LMS content without the separate need for registration and authentication. It also ensures that various functionalities are directly aligned with each user's specific role (such as administrator and regular user) and learning needs. As the LMS developed in this project is going to be a part of the SalesCue application, it will already include the HR integration by default. This efficiently removes the need for additional setup or configuration, ensuring a more effective learning experience from day one.

Ellis (2009, 2) explains that LMS **administrative tools** enable administrators or educators to handle user registrations, define roles, and set curricula. Admins create and distribute the course content and supervise assessments. Administrative tools also allow for comprehensive reporting on user

performance and help schedule resources and activities for learners and instructors. These functionalities ensure that educational programs are not only well-organized but also aligned with organizational compliance and certification needs. (Mts.) Sjørebø et al. (2009) emphasizes that an educator's motivation and engagement are crucial for maximizing student utilization and benefit from e-learning systems.

As mentioned above the administrator of a certain SalesCue Workspace is also the administrator of the LMS. The goal is to provide them with powerful and easy-to-use tools to create engaging content for the learners. Admins will be able to control access to various courses and they will have tools to set prerequisites for courses, thereby defining a structured learning path for users. This feature ensures that learners meet the necessary criteria and are adequately prepared for more advanced topics as they progress. The LMS will also include comprehensive analytics features, enabling administrators to monitor and evaluate learner engagement and progress throughout the courses. Upon the completion of courses, admins will be able to issue certificates, adding a formal recognition element to the learning path.

Content **delivery** points out the format and way, how the content is presented to the learners. Learning might be led by an instructor or designed as individual self-study and based on the learning goals the content may consist of various multimedia, like image, video or text (Clark & Mayer, 2011). In the LMS for SalesCue platform the course walkthrough is intended as a self-paced learning journey, allowing users to progress at their own speed according to their individual schedules and learning needs. Users can get back to the courses that are under completion, however for quizzes, the administrator will be able to set a fixed time limit. LMS will support various types of files i.e. images, videos, PDFs and additionally admins can attach text content to a course, lesson or file level.

As Ellis (2009, pp. 3) states, “**Content development** encompasses authoring, maintaining, and storing the learning content”. Administrators will be able to publish a course they deem ready for the users to complete. However, it should also be possible to update the course content later with fresh materials. Admins will also be visually informed whether some of the course content files have been removed or updated in the SalesCue system.

Finally, the **Assessment capability** is an important part of any Learning Management System. Digital assessment tools are time-efficient, as quizzes and exams quickly measure students' understanding or knowledge. Students receive immediate feedback, allowing them to choose whether to move forward or revisit the current material. (Turner, 2015) In SalesCue LMS quizzes will accompany lessons in the course structure and their final shape will be up to administrators to decide. Admins will have several options to customize quizzes to better suit the needs of the company employees. The options include format of questions (single-choice or multiple-choice), potential time limit, number of questions to ask as well as pass percentage.

## 2.2 Why does a Sales Enablement Platform need an LMS?

Although the idea of supporting sales personnel and enhancing sales processes to increase deal closure dates back to the 1900s, the term *Sales Enablement* was only coined in the early 2000s. There are numerous definitions of the term, but most agree that *Sales Enablement* is a cross-functional discipline created to provide a company's sales team with the necessary tools, skills, training and processes to boost sales results and productivity. (Didner, 2019.)

A Sales Enablement Platform is a specialized software solution that facilitates this cross-functional discipline. It serves as a central hub where sales teams can access all necessary resources such as training materials, content, and tools to effectively engage with customers and prospects. These platforms typically offer integrations with other business tools, such as CRM systems, Digital asset management and Content editing services to streamline workflows and provide analytics that help measure the effectiveness of sales strategies. (SalesCue, 2023.) In the competitive business of Sales Enablement Platforms, the presence of an integrated Learning Management System can significantly influence a prospect's choice. While some competitors offer LMS capabilities, others do not, potentially placing them at a disadvantage.

To illustrate the goal behind integrating an LMS into the SalesCue product, the following excerpt from our product documentation captures the core objectives:

*As part of our vision to enable effortless selling, we aim to introduce the LMS tool to provide training and onboarding for dealers, partners and sales teams. This, in turn, will make*

*selling even more effortless as they will gain a better understanding of the company and its products. A straightforward and user-friendly approach to the LMS tool, accessible to even non-tech-savvy customers, should ensure an optimal customer experience. By offering the tool with customer branding and easy LMS course creation, it should be practical and deliver excellent time-to-value for our customers.*

Customers are increasingly seeking LMS solutions to efficiently train their staff. Implementing an LMS enables employees to get a deeper understanding of their products, sales tactics, and overall business strategies. When employees know their products well and understand the best sales techniques, they are more likely to succeed in meeting sales goals, which benefits the whole company. The LMS is tailored for professional customers who need education and onboarding solutions for their users, and who prefer a unified platform over using multiple systems. The intention is to provide a streamlined experience with a fixed price with no usage limitations. This means that once the LMS feature is activated on the SalesCue account, unlimited courses can be created, and all account users may participate as many times as needed.

To provide a clearer picture of how the LMS can be utilized within the SalesCue platform, I will now outline a couple of use cases. These examples will showcase how the LMS addresses specific needs of different user groups, from dealers to internal teams, improving both learning outcomes and business efficiency.

### **Use Case 1: Dealer Training**

Many large companies partner with hundreds of dealers who distribute their products but who are not a part of the company's internal operations. This separation can make training difficult and expensive. Implementing an LMS makes this process easier by allowing companies to guide their dealers through specific onboarding courses. This approach ensures that dealers have a thorough understanding of the products they are selling. Furthermore, the company can track the dealers' progress in these courses and certify them once they have completed their learning paths. An important observation is, that for the dealer training to take place, the LMS must be available also to the external users who may not have SalesCue credentials, granting them access to necessary training materials.

## Use Case 2: SalesCue Academy

SalesCue itself can use the LMS to establish an 'Academy' that provides courses on how to utilize SalesCue itself, alongside onboarding and sales training. This setup enables administrators to monitor who has completed specific SalesCue courses, ensuring that users are adequately prepared to use the SalesCue App in their sales activities. Upon completing these courses, users are awarded certificates, which they can share on platforms like LinkedIn, enhancing their professional profiles and validating their proficiency. This feature not only strengthens the LMS's upselling potential but also simplifies the onboarding process for the Customer Success team. For freemium users, this quick access to SalesCue knowledge can accelerate their understanding and usage of the platform, potentially leading to increased conversions through no-touch or low-touch upselling strategies. Additionally, this approach could replace the traditional 'Getting Started' package, offering a more dynamic and engaging entry point for new users.

When considering the integration of a LMS into our platform, two primary approaches were evaluated: purchasing an LMS from an external provider and developing our own solution. Each approach has its advantages and drawbacks. Purchasing an LMS could have provided a quick deployment and relied on the expertise of vendors specialized in educational technologies, potentially reducing the initial complexity of integration. However, purchasing LMS from a third party often involves higher long-term costs due to licensing fees, and there might be limitations in customization options, making it less flexible to specific needs or future scaling. Additionally, relying on external support for updates and bug fixes can lead to delays and less direct control over the system performance.

On the other hand, developing our own LMS allows us more control over both the content and the structure of the platform. By choosing to implement the LMS ourselves, we are aiming at designing a system tailored precisely to our operational needs and future development plans. This approach is expected to be more cost-effective, eliminating recurring licensing fees. Moreover, having control over the entire system architecture makes it easier to implement updates, develop new features, and quickly resolve any issues or bugs that arise. Furthermore, undertaking the development of our own LMS presents a unique and challenging project that not only serves as an

excellent topic for my thesis but also enhances our team's professional growth and skill development.

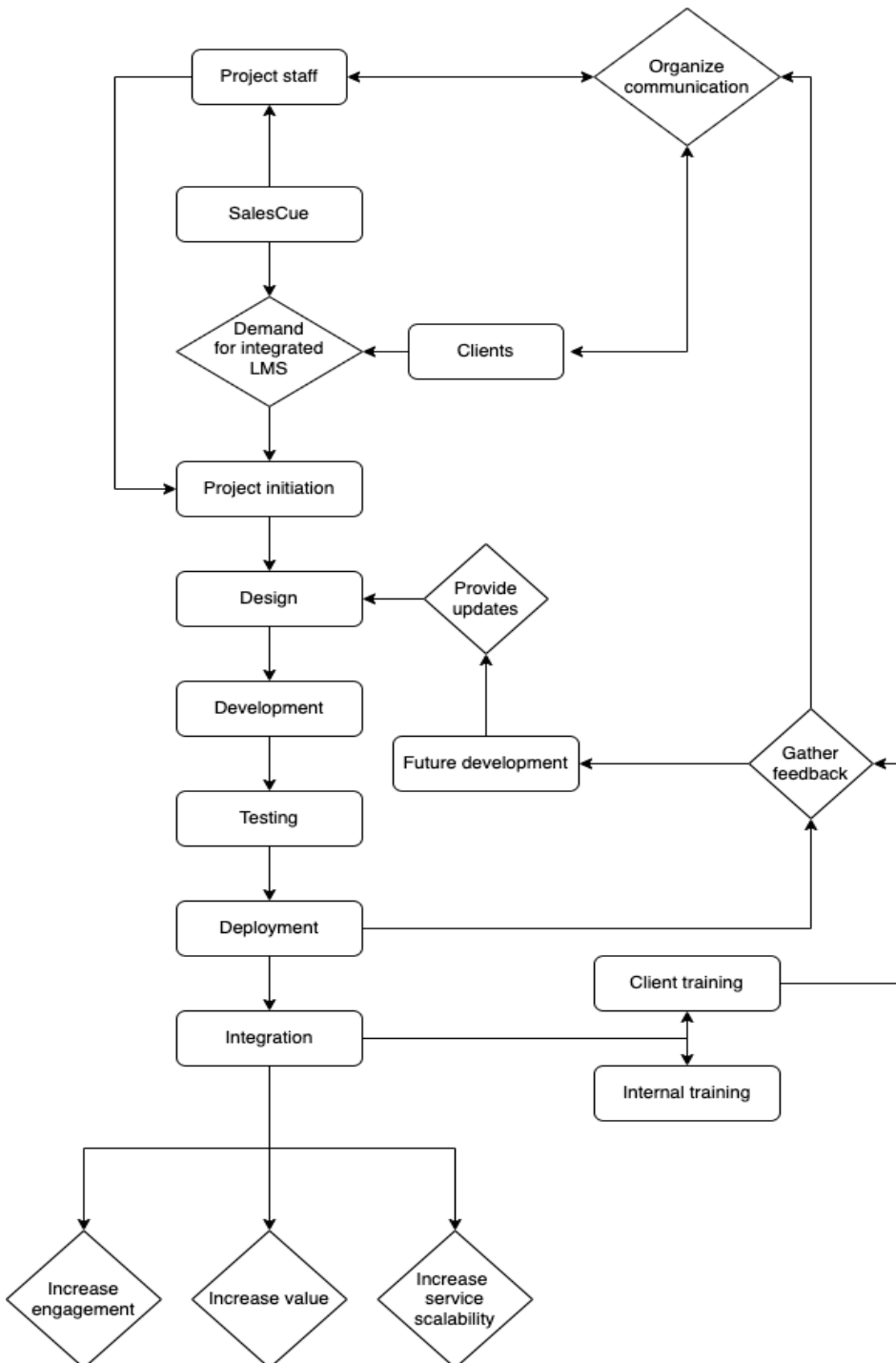


Figure 1: LMS project's processes and phases.

Ultimately, we decided to develop our own LMS. This decision was driven by the benefits of lower costs, greater flexibility in customization, direct control over the content, and the ability to quickly fix bugs and release updates. This approach ensures that the LMS grows and adapts to meet our goals.

The development of the LMS follows a structured process, making sure that each phase is executed with the purpose of delivering a high-quality product. The project begins with the identification of demand and progresses through several main stages before achieving full deployment and integration. Figure 1 above shows each phase.

The demand for the LMS is recognized by SalesCue, prompting the involvement of project staff who then initiates the project. This phase ensures that the project has a clear objective, and the necessary resources are allocated. Once the project is initiated, it moves into the Design phase. During this stage, detailed initiate plans and specifications for the LMS are created, addressing both functional and technical requirements. Following the design, the project enters the Development phase, where the actual coding and creation of the LMS take place.

After development, the LMS undergoes proper Testing to identify and fix any issues or bugs. This phase ensures the system is reliable. Upon successful testing, the LMS is deployed within the SalesCue platform, making it available for use. Post-deployment, the LMS is integrated into existing systems to ensure seamless functionality. This integration is supported by comprehensive internal training for our staff and customer training to help clients utilize the new system effectively. Training is an important step to ensure that users can leverage the LMS to its full potential. The integration phase aims to achieve three main goals. 1. By providing an interactive educational resource, we aim to increase user engagement and satisfaction. 2. The LMS adds significant value to our platform, offering clients a more comprehensive suite of tools. And 3. The LMS is designed to support growing numbers of users and expanding content, providing long-term viability.

After the LMS is deployed and integrated, we gather feedback from users and clients to assess its performance and identify areas for improvement. This feedback is important for identifying needs for future development and updates and making sure the LMS evolves to meet changing needs and user suggestions. In conclusion, the development of an integrated LMS within the SalesCue platform is a strategic move to improve our services and meet client demands. Through a structured workflow involving initiation, design, development, testing, deployment, integration, and continuous feedback, we aim to provide a high-quality, valuable, and scalable educational tool for our users.



## 2.3 Technological stack overview

In this chapter, I will introduce the main technologies used in the development of the LMS, which are MongoDB, tRPC, and React Native. I will present each of these technologies and explain the reasons why we chose them for this project.

**MongoDB** is an open-source document database that uses flexible and scalable NoSQL architecture. Released in 2009, it differs from traditional relational databases by storing data in JSON-like documents, which can have dynamic schemas known as BSON (Binary JSON). This approach helps integrate data in applications that are typically restricted by strict schema constraints found in relational databases.

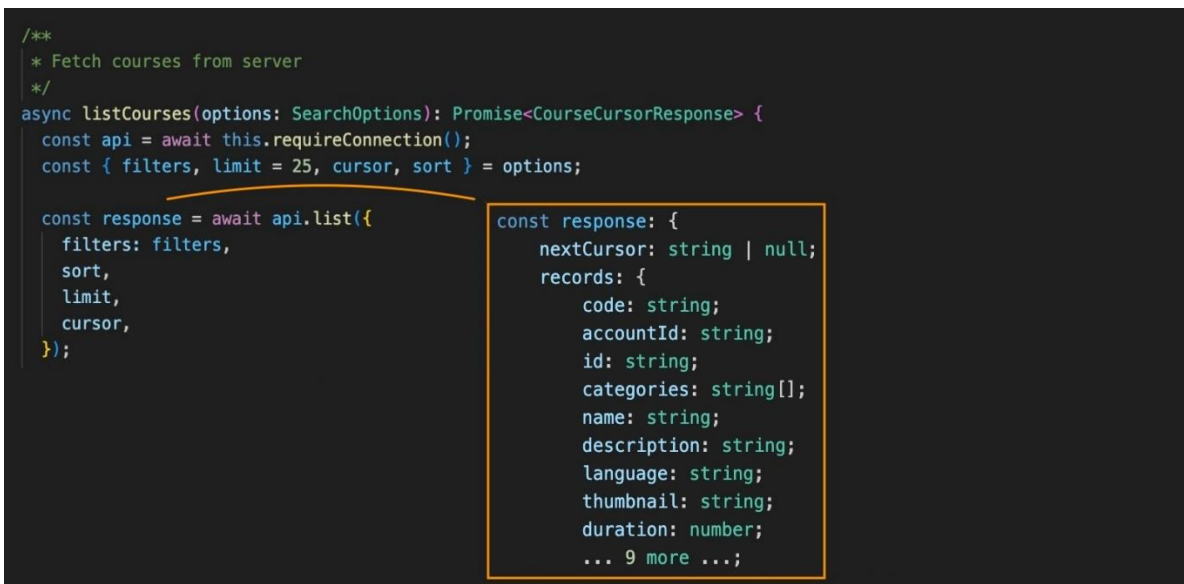
The document-oriented storage system of MongoDB allows fields to vary between documents and data structure to be modified over time. This is particularly useful for applications that require a quick adaptation to changing requirements. Unlike relational databases that store data in tables and rows, MongoDB's documents more closely resemble the structure of objects in programming languages, which can simplify how developers write and think about their database code. Scalability features like sharding, which distributes data across a cluster of machines, and replica sets, which ensure high availability and disaster recovery make MongoDB suitable for enterprise-level applications that require handling vast amounts of distributed data. (MongoDb, 2024.)

In the context of our LMS project, MongoDB's flexible schema is important for containing the diverse and evolving data types associated with educational content. Whether tracking complex user progress data or storing multimedia teaching materials, MongoDB enables the LMS to scale efficiently and adapt quickly to new educational methodologies or user feedback. Additionally, the ability to easily integrate MongoDB with other components of our tech stack, such as React Native and tRPC, enhances the development process and improves the overall performance and responsiveness of the LMS.

**tRPC**, a short for TypeScript Remote Procedure Call, is a modern framework designed to reinforce the development of type-safe APIs by utilizing the capabilities of TypeScript and eliminating the need for schemas and manual code generation. Due to TypeScript, the type safety is ensured

throughout the entire stack, thus providing a seamless integration between the client and server sides of an application.

tRPC's main advantage is its ability to provide end-to-end type safety in application development, which allows developers to maintain a consistent type definition across the front and back ends, which is automatically updated when changes occur in the codebase. This helps reduce bugs and makes the development process easier, as the developer can always see the automatically inferred types from backend to the frontend without manual definitions, as is indicated on the following figure:



```
/**
 * Fetch courses from server
 */
async listCourses(options: SearchOptions): Promise<CourseCursorResponse> {
  const api = await this.requireConnection();
  const { filters, limit = 25, cursor, sort } = options;

  const response = await api.list({
    filters: filters,
    sort,
    limit,
    cursor,
  });
}
```

```
const response: {
  nextCursor: string | null;
  records: {
    code: string;
    accountId: string;
    id: string;
    categories: string[];
    name: string;
    description: string;
    language: string;
    thumbnail: string;
    duration: number;
    ... 9 more ...;
  };
}
```

Figure 2: Fetching response from the API call automatically offers its type structure.

Another benefit of tRPC is its simplicity and improved developer experience. It integrates with modern development stacks and offers intuitive hooks and utilities that make data fetching and server interaction simple. tRPC supports multiple transport layers, including HTTP and Web Sockets, making it suitable for a wide range of applications. tRPC is optimal in environments, where maintaining type control across the full stack of an application is most important. It is well-suited for full-stack TypeScript applications, real-time web applications, and projects that have quick development cycles. Its capability to handle real-time data makes it a good choice for interactive dashboards, live chat systems, and collaborative tools, where user interactions require immediate updates without risking type safety. Its ease of use and elimination of boilerplate code makes tRPC

a good option for prototyping, allowing teams to quickly test and iterate on new ideas. (tRPC, 2024.)

As already mentioned, tRPC has recently been integrated into our technology stack. Apart from the LMS, it is currently being implemented in another project that is also under development. Our architects have determined that tRPC is the best solution for the LMS due to its efficient and robust handling of server-client communication. The final assessment of tRPC's suitability will be conducted after the deployment of the LMS, allowing us to evaluate its performance and effectiveness in a live environment.

SalesCue platform started as an iOS application and is built with **React Native** frontend. React Native (RN) is a mobile application framework developed and maintained by Meta and the open-source community. It allows developers to build cross-platform mobile apps using JavaScript/TypeScript and React, offering both developer efficiency and native performance. React Native enables the creation of applications for both iOS and Android platforms using a single codebase. SalesCue application utilizes React Native for Windows and React Native for Web extensions to enable the app to run also on Windows devices and web browsers. While LMS is initially being developed only for the web platform, with RN it shouldn't be too difficult to later enable the LMS to other platforms as well, with minimal code changes.

React Native interacts directly with mobile platform APIs and rendering by using native components, ensuring that applications perform like native apps while integrating with the corresponding operating system. The framework uses React library, which means that developers can utilize a vast number of React components. Moreover, RN introduces features like hot reloading that allows developers to see the results of their latest code changes in real time without restarting the app. RN also benefits from an active community. The framework is supported by numerous third-party plugins, libraries, and tools that extend its capabilities beyond those provided by the framework itself. (React Native, 2024.) React Native is advantageous in scenarios where businesses need to develop mobile applications efficiently without compromising on the user experience. Ideal projects would be consumer-facing apps that demand high performance, smooth animations, and native look and feel, such as sales platforms, social media apps, and entertainment applications.

In addition to the primary technologies driving the development of our Learning Management System, the infrastructure and security aspects are equally important for ensuring the reliability of the platform. The entire project is hosted on Amazon Web Services (AWS), which provides a secure and scalable cloud environment. On the security front, we utilize JSON Web Tokens (JWT) for secure authentication and authorization. JWTs help keep the information safe while being transmitted as JSON objects, protecting our users' data from unauthorized access.

## **2.4 UX and interface design**

A user-friendly interface is essential for any application and especially for an educational tool like a Learning Management System. A well-designed interface enhances learning by making information accessible and easy to navigate, which is necessary for users to effectively gain knowledge. Moreover, an intuitive and engaging interface encourages users to continue using the app, significantly increasing user satisfaction and retention rates. At SalesCue, our designers are committed to following best practices in user experience (UX) design, ensuring that every aspect of the LMS is optimized for user engagement and ease of use. It's also vital that the LMS reflects the distinctive SalesCue look and feel, maintaining brand consistency across all touchpoints. This consistency helps in reinforcing user trust and improving the overall experience. To achieve this, our design process involves the use of Figma, a collaborative design tool where SalesCue's UX/UI designs are stored and continually refined. This platform allows our team to update and improve design elements dynamically, ensuring that our LMS always meets the highest standards of design quality.

In the following section, I will present a few examples of the user interface designs implemented in the LMS. These examples are the starting point of frontend development, but the design also frequently changes based on feedback from the actual usage of various components.

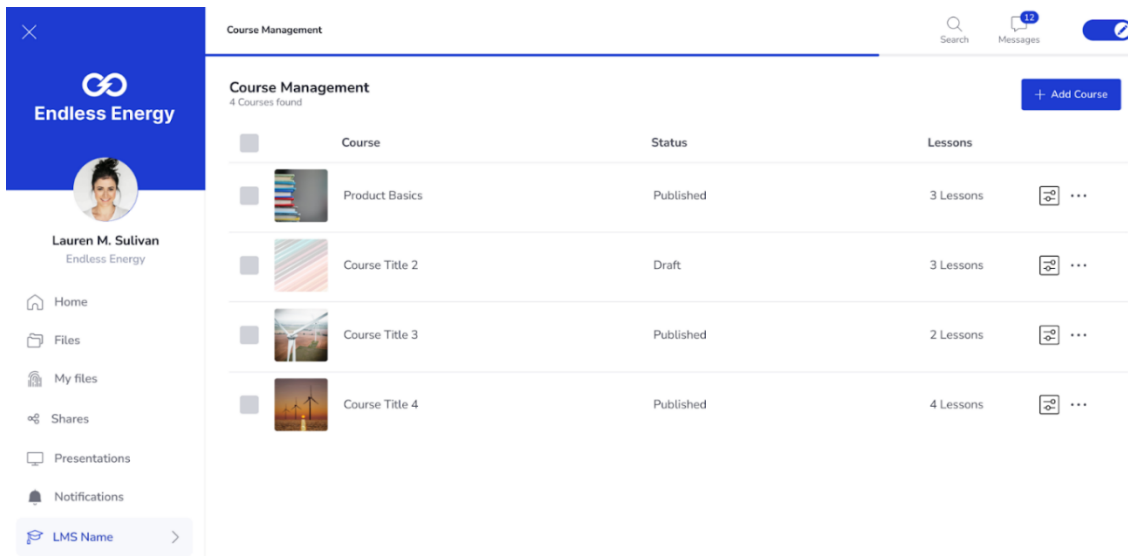


Figure 3: Course Management View in Figma

The Figure 3 above represents the Course Management interface of LMS, which serves as a central hub for administrators to oversee and manage courses. This view is designed to be intuitive, providing easy navigation and operation. It lists courses with essential details such as course titles, their publication status, and the number of lessons each course contains. The interface aims at a clean and organized layout, with a straightforward visual hierarchy. Actions like adding a new course or editing existing ones should be easily accessible and provide an intuitive management experience.

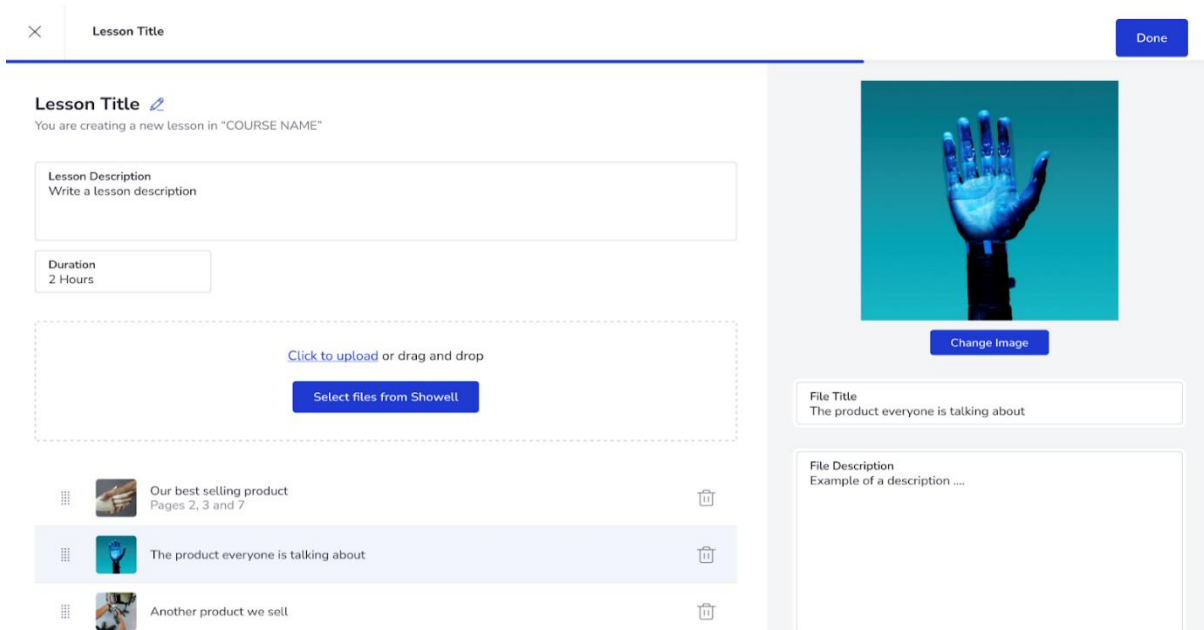


Figure 4: Lesson editing View in Figma

The Figure 4 showcases the "Lesson Creation" interface, designed to streamline the process of setting up a new lesson. At the top, users can enter a lesson title and provide a detailed description, which clearly communicates the lesson's purpose to learners. Below that, admins are asked to specify the lesson's duration. For adding multimedia content, the interface includes both a drag-and-drop area and a button to select files from SalesCue, accommodating different user preferences for uploading resources. The area on the right shows the selected file's details and offers options to edit file's title, description, or thumbnail. The admin can also subsequently change the order of the lesson's content, discard some of the files or add new ones.

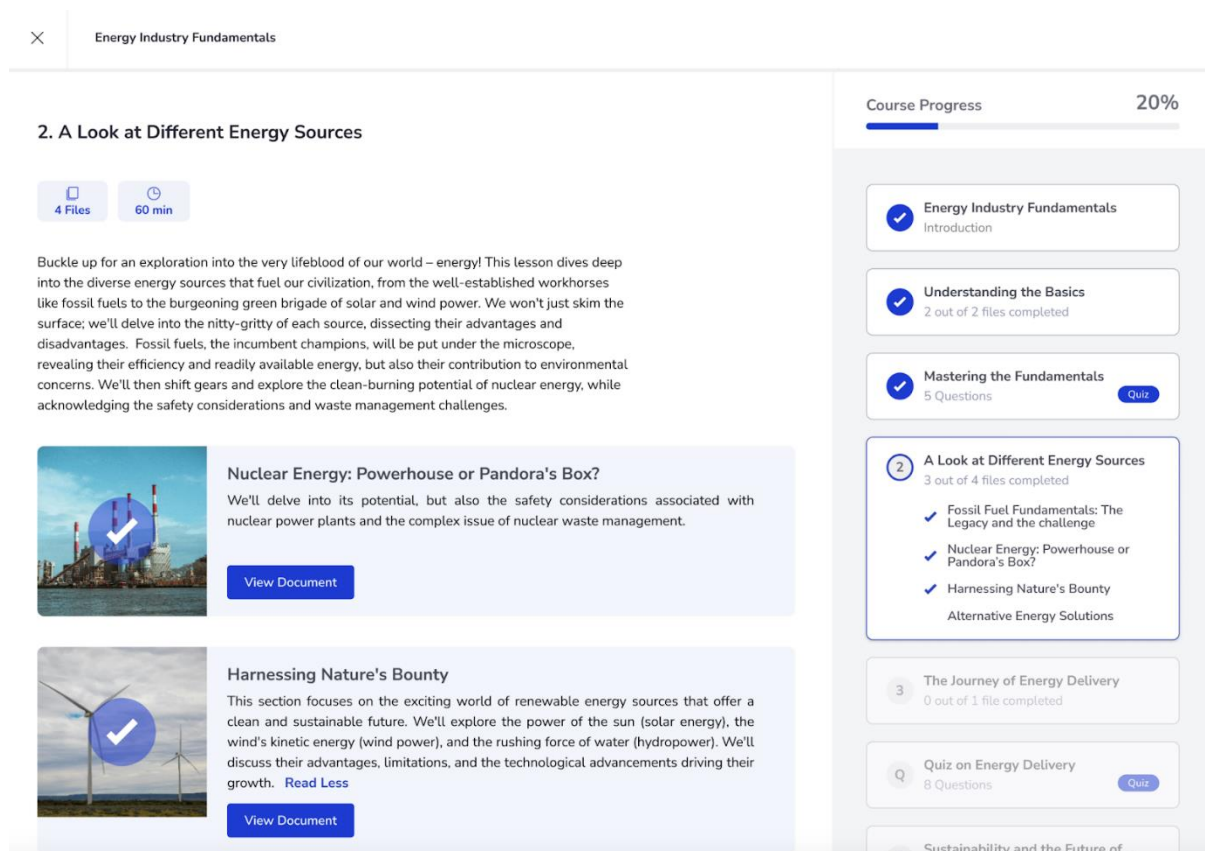


Figure 5: Course completion view in Figma.

The Figure 5 displays Course Completion View, the most important part of participant's interface in the LMS. This view has been specifically designed to enhance the learning experience by focusing on clarity and usability. On the right side, a sidebar navigates through different sections of the course, highlighting the progression of completed versus not-completed materials. Each section is clearly marked with check marks when completed, providing learners with a visual representation of their progress. This design ensures that participants can easily see how much of the course they have completed, with the overall course progress percentage displayed at the top of the sidebar.

The main content area on the left presents detailed lesson descriptions and associated materials which participants can access to deepen their understanding of each topic. While the course administration views utilize many of the ready-made SalesCue's common components, this course completion interface has been newly developed specifically for the LMS.

This chapter has explored the extensive and detailed user interface designs for the LMS, as documented in Figma. These designs not only cover desktop interfaces but also extend to small mobile phone screens, ensuring that the system is fully responsive and accessible on any device. From the developer's point of view, it's much easier and faster to implement the frontend when such comprehensive designs are available. The designs are overall categorized from two perspectives: the administrator's (management) and the user's (completion) point of view. This dual structure reflects how the system is set up and allows us to customize the features and interfaces for different roles in the LMS.

It is important to mention, that the design process is dynamic and ongoing. As development and testing progresses, new insights often arise and revisions and updates to the initial designs are needed. This iterative process ensures that the LMS remains user-centric and responds to the real-world usage challenges as they occur.

### 3 Implementation

This chapter provides a detailed overview of the implementation process of the Learning Management System into the SalesCue platform, highlighting the main components such as database design, backend and frontend development. The accompanied diagram depicts various layers of the whole system development.

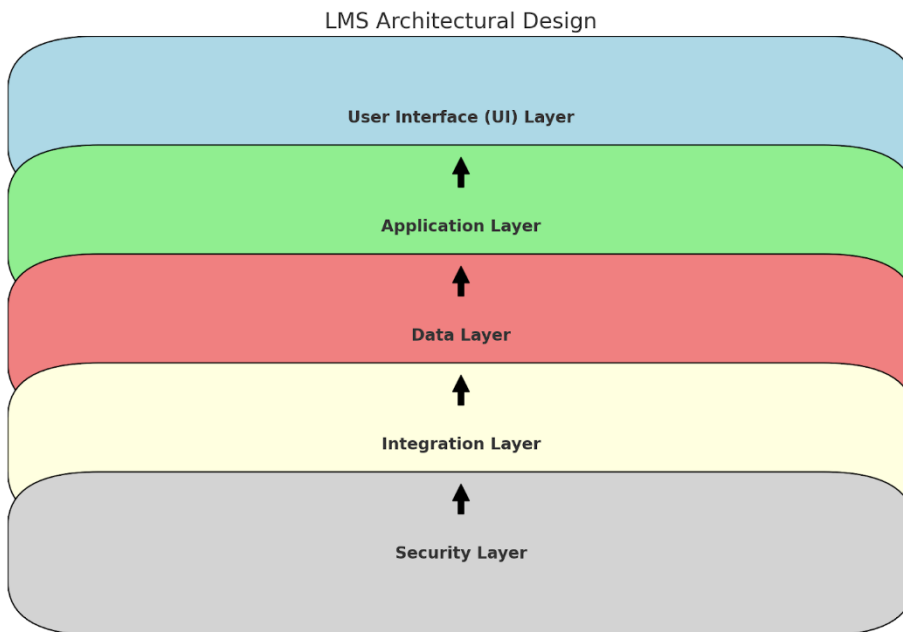


Figure 6: LMS Architectural design

The Database design chapter, represented by the Data Layer, describes the structure of the LMS's data storage, focusing on the collections in MongoDB and their mutual relationships. It creates a foundation for the subsequent Application Layer, covered in detail in the Backend development section, which focuses on the backend services using tRPC, explaining the implementation of the business logic and API endpoints.

Moving up the stack, the Frontend Development section, aligned with the User Interface (UI) Layer in the diagram, discusses the creation of the user interface using React Native. This section covers how data flows from the backend to the frontend, ensuring synchronization and an engaging user experience. The UI Layer handles all user interactions, providing a responsive and interactive interface.



For the purpose of integrity, the diagram also includes a Security Layer and an Integration Layer. The Security Layer ensures robust data protection and user privacy, utilizing mechanisms such as JWT tokens for authentication and role-based access control. The Integration Layer manages interactions between the LMS and other SalesCue systems and packages. However, detailed discussion of these layers is beyond the scope of this thesis and my work, so they will not be covered in depth.

### 3.1 Database design

The Learning Management System database design is based on five collections in MongoDB: **course**, **lesson**, **participant**, **category**, and **certificate**. Each collection stores specific types of documents with unique structures and relationships to other collections (Figure 7). For simplicity, the fields pointing out to dates (*createdAt*, *modifiedAt*, *removedAt*) are left out of the scheme, but they exist in each collection. The field *removedAt* is used for the so called ‘soft’ delete, meaning that the entity is marked as removed, but not physically removed from the database. This comes in useful if the user decides to restore a removed object.

#### COURSE

The foundation stone of the LMS scheme is the course collection, which stores documents representing individual courses. As well as all other documents, it has a unique *id*, which is an ObjectId generated by Mongo. Each course has a *name*, *description*, that should provide a detailed overview of the course content, and a *language*, which specifies the language in which the course is offered. The *active* field is a boolean flag indicating, whether the course has been marked by an administrator as available to users. A *thumbnail* field holds the path to an image that serves as the course’s thumbnail, providing a visual representation of an individual course. The *duration* field indicates the total time required to complete the course and is calculated based on its content durations.

The *categories* field contains an array of ObjectIds, which point to the category collection, linking the course to the relevant categories. The *lessons* field is also an array of ObjectIds, representing the lessons included in the course. Lessons are therefore the content of each course. The *certificate* is an optional field, which may hold an identifier linking it to a certificate awarded upon course completion. Each course has a several fields that specify who has access to the course: *groups* array consists of groups, that have gotten an access to the course and similarly, the *users*

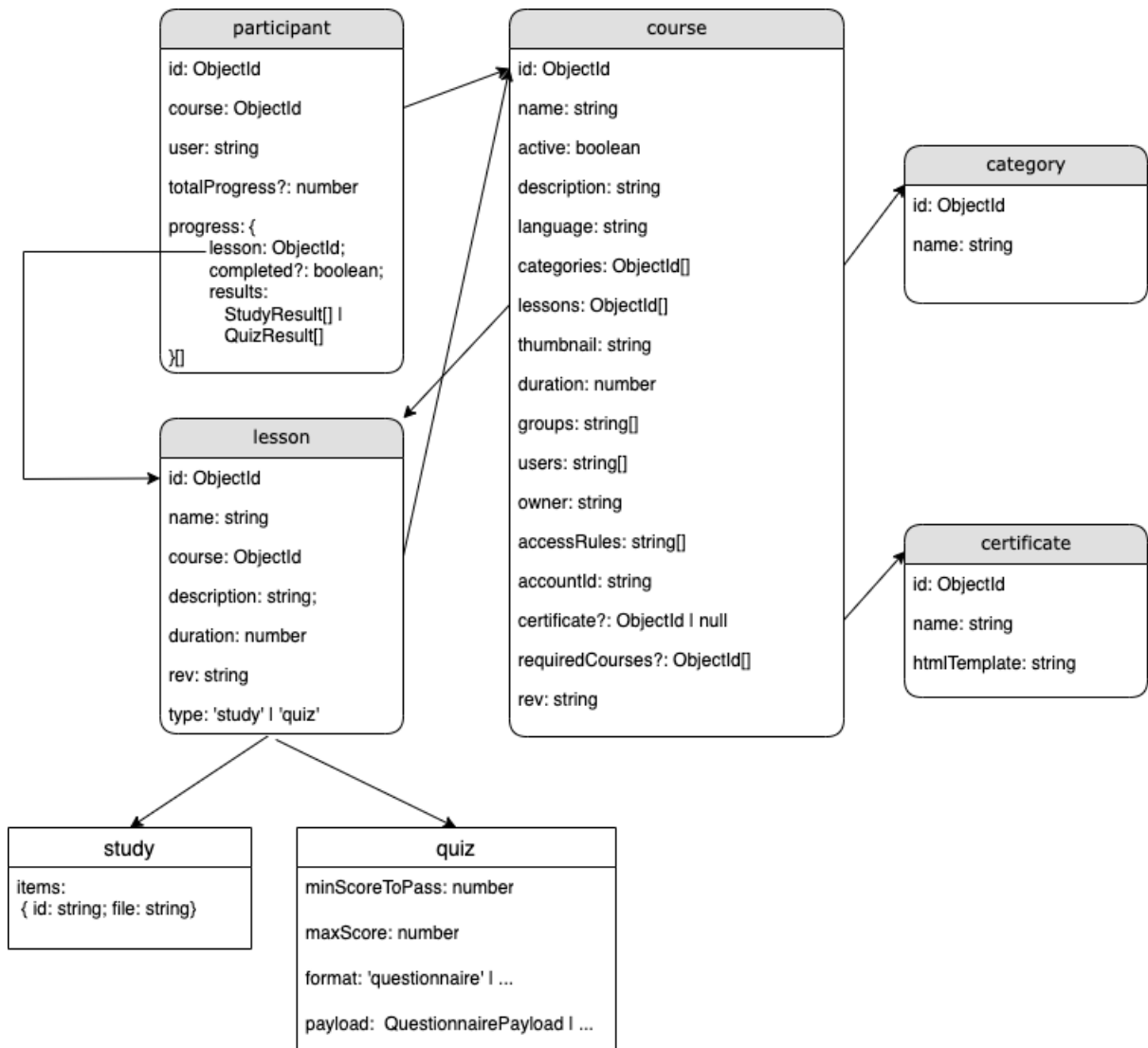


Figure 7: LMS Database scheme.

array contains users that have access to the course. Both groups and users are represented by their ids in the SalesCue system. The *accessRules* field serves as a mechanism to control access to the course. It is essentially an array that contains identifiers (such as account IDs, group or user IDs), that define, who has access to the course. By specifying these rules, the system can ensure that only authorized users, groups or accounts can view and interact with the course content. The

*owner* field identifies the user, that owns the course and the *accountId* associates the course with a particular account, linking it to a broader organizational structure.

The *requiredCourses* field lists *ObjectIds* of prerequisite courses, that must be completed before enrolling the current course, ensuring proper learning progression. Finally, the *rev* field serves as a revision identifier for version control, which is updated whenever the course is modified in the database. This approach is more effective than using simply the *modifiedAt* timestamp because there can be multiple changes at the exact same time.

## LESSON

The lesson collection in the LMS database stores documents, that represent the content of a course, with a course containing one or several lessons. There is no limit to the number of lessons a course can have. Like the course document, the lesson document includes fields such as *id*, *name*, *description*, *duration*, and *rev*. These fields serve to uniquely identify the lesson, provide a brief overview, approximate the time required to complete the lesson, and manage version control with the *rev* field.

A crucial field in the lesson document is the *course* field, which is an *ObjectId* pointing to a specific course. This means that each lesson is linked to one course only, establishing a clear hierarchical structure between courses and their respective lessons. The *type* field categorizes lessons into two types: **study** and **quiz**. Each type has additional fields tailored to its specific content and purpose.

For a study lesson, the document includes an *items* array, which contains files that constitute the lesson content. These files are the materials that users need to review in order to complete the lesson. In contrast, a quiz lesson includes properties, that are specific to quiz content and assessment. These fields are *minScoreToPass* and *maxScore*, which set the scoring parameters for passing the quiz. The quiz document includes a *format* field, which can be a 'questionnaire' or other potential formats. This allows the quiz to be more versatile in the future, accommodating different types of assessment formats beyond the traditional question-and-answer style.

The *payload* field within a quiz document contains the specific content for the quiz, defined as *QuestionnairePayload* or other types, depending on the quiz format. The *QuestionnairePayload* includes the questions and associated data, such as answer options, the correct answers, and various settings related to the questionnaire, for example the number of questions to be asked. Essentially, the payload serves as the content of the quiz, outlining all necessary information for its execution.

## **PARTICIPANT**

The participant document in the LMS database schema represents a user's enrollment and progress within a specific course. Each participant is uniquely identified by a combination of the *user* field, which points to a SalesCue user ID, and the *course* field, which is an ObjectId referencing a particular course. This structure ensures that there is one participant record per user per course. The participant document includes a *progress* array, where each object within the array corresponds to a specific lesson. Each progress object contains a *lesson* field, which is an ObjectId pointing to the respective lesson, ensuring that there is only one progress record per lesson for each participant. The *completed* field is a boolean flag indicating whether the lesson has been completed.

Additionally, the *progress* object contains a *results* array, which varies based on the type of lesson. For study lessons, the results array consists of the content items that have been viewed by the participant, tracking their engagement with the lesson materials. For quiz lessons, the results array includes information on the answered questions, capturing the participant's responses and performance on the quiz.

The category and certificate collections are relatively straightforward but play essential roles in organizing and enhancing the functionality of the LMS. The category document is used to classify courses into different groups or topics, making it easier for users to find and enroll in courses that match their needs. In the course document, the *categories* field is an array of ObjectIds that link to multiple category documents. This relationship allows a single course to belong to multiple categories, providing flexibility in course organization and filtering.

The certificate document represents the certification awarded to participants upon completing a course. It has a *htmlTemplate* field, representing the template used to generate the certificate. The course's *certificate* field is optional, meaning that not all courses necessarily award certificate.

To summarize, the database schema for the LMS manages the relationships between courses, lessons, participants, categories, and certificates. Each course can contain multiple lessons through an array of lesson ObjectIds. Courses can belong to multiple categories, and may award a certificate upon completion, linking to a certificate document. The lesson collection links directly to a specific course, with each lesson being part of only one course. Lessons are categorized into study or quiz types, with study lessons containing content items and quiz lessons including scoring parameters and a flexible payload for various formats. The participant collection tracks user enrollment and progress within courses, linking each participant to a specific user and course. It records detailed progress for each lesson, including completion status and results, which vary based on the lesson type.

Throughout the development process, the database schema has evolved to meet new needs and insights. A significant change was the separation of quiz lessons from study lessons into distinct types, reflecting their unique requirements. Such adjustments are a normal part of development and may continue to happen as new requirements arise. However, these changes should stabilize once the database is populated with real data, so that the data remains consistent and reliable.

## **3.2 Backend development**

The backend of the LMS handles data storage, business logic, user authentication, and communication between the client-side application and the server. In this chapter I will delve into the key components of the backend architecture. It starts with an overview of the project configuration and tRPC. After that I will address on the database connectivity and configuration of various routers. Finally, I will focus on routers' endpoints in more detail.

In the SalesCue LMS project, the serverless framework is used to manage and deploy backend services to AWS. Serverless's benefits are its easy configuration, fast deployment and the fact that it is cost-efficient. It automatically scales with the number of incoming requests and the charging is based on actual usage, which reduces costs compared to traditional server-based architectures. Its

main purpose is to deploy code to AWS Lambda. The *serverless.yml* configuration file details the setup and deployment instructions. It includes environment variables defining MongoDB connection strings, user credentials, JWT secrets, and other configurations for production and development environments. The provider configuration specifies AWS as the cloud provider, including settings for the runtime environment, region, and IAM roles for necessary permissions.

The tRPC serves as the project's backbone, as it allows to create a router and procedures that handle various API requests. First, the tRPC needs to be initialized with an appropriate context. The context provides a way to pass information (like user data and database connections) to all resolvers. The `createContext` function is responsible for initializing the context for each request. The function extracts the JWT token from the request headers, verifies it, and decodes the user identity. This process ensures that only authorized users can access the backend services and that the user information is reliable. After that, the tRPC instance is initialized using `initTRPC` method, imported from the tRPC server package:

```
import { initTRPC } from '@trpc/server';
import { Context } from '../context/context';

export const t = initTRPC.context<Context>().create({
  errorFormatter: function errorFormatter(opts) {
    ...
  },
});
export const router = t.router;
export const publicProcedure = t.procedure;
```

Figure 8: tRPC initialization.

The `router` and `publicProcedure` exports are used to define API endpoints. The `router` function creates a new router instance, and `publicProcedure` is a shorthand for creating endpoints, that are accessible without authentication. These are the building blocks for defining the API structure and endpoints.

The next step is to get access to the MongoDB database hosted on MongoDB Atlas. The connection is managed through a function that retrieves the necessary database URI and user credentials from environment variables and AWS Secrets Manager. This approach ensures, that the sensitive

information, such as database passwords are accessed securely and only when needed. Once the connection parameters are fetched, the function establishes a connection to the MongoDB database. This connection is reused for all subsequent database operations.

Using tRPC, we can create routers that handle various API requests. The router setup begins with defining the `apiRouter`, which is the main entry point for the API, and its sub-routers for different functional areas such as courses, lessons, and categories:

```
import { router } from '../server/trpc';

export const apiRouter = router({
  v1: router({
    course: courseRouterV1,
    lesson: lessonRouterV1,
    category: categoryRouterV1,
  }),
});

export type ApiRouter = typeof apiRouter;
```

Figure 9: tRPC API router.

The `apiRouter` contains a nested `v1` router, which includes the sub-routers responsible for handling the endpoints related to its corresponding area. API versioning is important for maintaining the backend services without disrupting existing clients. This allows to introduce new features, improvements, and bug fixes without breaking existing clients that rely on the older version of the API.

The course, lesson, and category routers roughly correspond to the database models, but do not include a distinct participant router. Instead, functions related to participants, such as `joinCourse` or `updateLessonProgress` are integrated within the course and lesson routers, as those operations are also tied to courses and lessons. The certificate functionality has not been implemented at the time of writing this thesis, but there might be a new router for it in the future. Generally, the routers include endpoints related to basic CRUD operations, as listed in the table below:

	CREATE	READ	UPDATE	DELETE
course	create, join	list, listOngoing, listParticipants, getParticipant	update, grantAccess, re- moveAccess, remove, leave, updateLessonOrder	delete
lesson	createStudy, createQuiz	list, getLesson,	updateStudy, updateQuiz, completeQuiz, remove, up- dateQuizProgress, updateS- tudyProgress	delete
category	create	list, listNonEmpty	update, remove	delete

Table 1: Course, lesson, and category router endpoints.

Some endpoints, such as `create`, `list` and `delete` are self-explanatory, however, other endpoints deserve brief explanation. `remove` endpoint is generally listed under the 'update' category, because it performs soft remove, updating the course with a `removedAt` timestamp, rather than deleting it permanently from the database. The `join` method creates a new participant record and `leave` updates the participant, marking it with a `leftAt` timestamp. `listOngoing` endpoint within the course router lists all ongoing courses for a specific participant and `listParticipant` retrieves all participant for administrative purposes, such as analytics. The `grantAccess` and `removeAccess` methods update course access rights. These are distinct from the `update` method because the logic for checking and modifying access is quite complex. The `updateLessonOrder` endpoint also updates the course, specifically the order of the `lessons` array. The order is vital as it determines the sequence in which participants are presented with course content.

The lesson router has parallel methods for handling two types of lessons, quiz and study, such as `createStudy`, `createQuiz`, `updateStudy`, and `updateQuiz`. The `updateStudyProgress`



and `updateQuizProgress` endpoints update a participant's progress array with the relevant results, and the `completeQuiz` method assesses the participant's quiz results. Finally, the category router's `listNonEmpty` endpoint returns only categories, that are in use within some course, filtering out empty categories, not being used anywhere.

```
list: publicProcedure
  .input(
    z
    .object({
      filters: z.array(zFilter).optional(),
      sort: z.array(zSortMode).optional(),
      limit: z
        .number()
        .gt(0)
        .lte(parseInt(process.env['max_list_limit'] as string)),
      cursor: z.string().nullable().optional(),
    })
    .strict()
  )
```

Figure 10: Course router, list method's input.

As the final topic of the backend development chapter, I will examine in more detail course router's endpoint `list`, as an example of how endpoints are constructed in the LMS backend. This method handles various functionalities such as input validation, database querying, filtering, sorting, and pagination. In the Figure 10 above, Zod is used to validate the input parameters for the endpoint. Zod is a TypeScript-first schema declaration and validation library, and it verifies the input at runtime. The input consists of filters, that can restrict the query, sort options, limit for the maximum number of found courses and an optional cursor for pagination.

After the input has been validated, the connection to the MongoDB is made and a query is constructed using several criteria:

```
const query = {
  $and: [
    ...(input.filters?.map(filter =>
      parseFilterToQuery(filter as Filter)
    ) ?? []),
    ...(context.identity.scope === 'user' ? [{ active: true }] : []),
    { removedAt: null },
    { accountId: context.identity.accountId },
    getCourseAccessFilter(context.identity),
  ],
};
```

Figure 11: List method, the formation of query.

filters array is mapped to MongoDB query filters, only active courses are included for users (as opposed to admins) and only courses that are not marked as removed are included. Additionally, courses must match caller's account ID and accessRules array, which is generated based on the caller's identity. After the query has been created the result is retrieved from the database using find method:

```
db.collection<Course>('course').find(query)
```

The result is first sorted based on the provided sort modes and cursor-based pagination is implemented by using the limit and cursor parameters. The cursor is used to skip documents and fetch the next set of results. Finally, the following result is return from the list endpoint:

```
const preparedCourses = await prepareCourses(courses, context.identity);

const result: ListResult<typeof preparedCourses> = {
  records: preparedCourses,
  totalRecords,
  nextCursor: nextCursor ? nextCursor.toString() : null,
};

return result;
```

Figure 12: List method, return statement.

The retrieved courses are processed in the prepareCourses function, which has two main tasks. First, it parses the course using the zCourse scheme, which modifies certain properties' types, such

as changing Mongo's ObjectIds into strings, so that their type is consistent with the frontend. And second, if the caller is not admin, the function omits some unnecessary properties, for example information about course's access rules. `totalRecords` is the total number of records that match the query and `nextCursor` points to the next set of results, if applicable.

The routers and endpoints are designed to securely fetch data, ensuring it is appropriately filtered and formatted based on the user's role. The data is returned in a user-friendly format, making it ready for use in the frontend application. Additionally, tRPC provides important type safety, securing that the client and server remain in sync and reducing the risk of runtime errors.

### 3.3 Frontend development

The frontend development of the LMS is a critical component of the project, as it defines the user experience and interaction with the system. This chapter discusses the technologies, data management strategies, and key components that form the LMS frontend.

```
const client = createTRPCProxyClient<ApiRouter>({
  links: [
    httpBatchLink({
      url: 'https://api.example.com/',
      headers: {
        Authorization: 'Bearer ' + lmsToken,
      },
    }),
  ],
});

return client;
```

Figure 13: tRPC client initialization.

In the SalesCue platform, services contain the core business logic and handle data retrieval and caching. Services are designed to mirror the backend structure, including Course Service, Lesson Service and Category Service. The services use a tRPC client to connect to the tRPC backend. The code in the Figure 13 above demonstrates how the tRPC client is initialized. The client is then available in all related services and offers access to all the API endpoints.

In the SalesCue platform, services handle frontend data management and are used primarily through two methods: data source hooks and actions. Data source hooks connect a service with a React component. They manage data fetching and ensure that the data remains up-to-date by reacting to events such as create, update, and remove.

```
const useCourses = createDataSource(
  'courseService',
  (service, params: Query, cursor, limit) =>
    service.listCourses({
      ...(params as { filters: FieldFilter[] }),
      cursor,
      limit,
    }),
  {
    shouldRefresh: ({ created = [], updated = [], removed = [] }) =>
      Boolean(created.length) ||
      Boolean(updated.length) ||
      Boolean(removed.length),
  }
);

const {
  records: courses,
  fetchNext,
  isFetched,
  isFetching,
  totalRecords,
} = useCourses(query, { limit: 10 });
```

Figure 14: Custom React hook useCourses.

The useCourses hook (Figure 14) is then used in components to fetch and display course data. This ensures that the component data is always current and provides other useful information, such as whether the data is still being fetched. If `isFetched` is true, then the courses can be rendered, otherwise the component shows loading skeleton.

```
export async function updateCourse(id: string, specs: UpdateSpecs) {
  const intl = getIntl();
  await runAction(
    async () => {
      await getCourseService().updateCourse(id, specs);
    },
    { successMessage: intl.formatMessage(messages.courseSaved) }
  );
}
```

Figure 15: Action method to update a course.

Actions are used mostly for single operations triggered by user interactions, such as creating or updating a course (Figure 15). These actions have a built-in error handling to simplify development. The `runAction` method will display an error message to the user, if the action fails. Conversely, if the action is successful a toast with a success message is flashed.

In the SalesCue platform, services are also important for providing caching mechanism to store fetched data locally and manage it efficiently. Whenever data, such as a course or a lesson, is fetched from server, it is stored in a local cache object within the corresponding service, indexed by its ID. When data is requested, the service first checks if it already exists in the cache. If the data is present, it is returned directly from the cache, avoiding an unnecessary server call. If the data is not found in the cache, it is fetched from the server and then stored in the cache for future use.

Next, I will focus on the user interfaces I have developed using React Native, based on the designs in Figma. They can be divided into two categories in general based on the user role. They are either the management and edit views for administrators, or course completion views for regular users.

The admin views are designed to enable course and lesson management and the two main screens are `CourseManagement` and `LessonManagement`. These screens enable administrators to create, update, and manage courses and lessons using interfaces that are visually similar to other management screens in the SalesCue application. The `CourseManagement` screen is a pivotal view for administrators, providing a comprehensive list of all courses. The courses are fetched in batches of 15 to optimize performance and load times, with additional courses fetched as the user scrolls down. The screen displays a vertical list of courses, each represented in a row. Each row includes the course thumbnail, name, number of lessons, and the total duration of the course (Figure 16). Additionally, each course row includes a *Properties* button that opens a side panel, displaying detailed course information and allowing administrators to edit course specifics. A prominent *Add Course* button on the screen opens a modal view for creating a new course.

	Course ▾	Category	Lessons	Duration	Created ▾		
<input type="checkbox"/>	 Basics of Robotics	Product	5 lessons	2 h 15 min	Apr 25, 2024	...	

Figure 16: Course row.

When a course row is pressed, it navigates to the LessonManagement screen, so it is easy for users to transition between managing courses and their respective lessons. The LessonManagement screen is similar to the CourseManagement and contains a list of the course's lessons, which can be either a study or a quiz. The quiz and study lessons are visually distinct from each other, making it easy for the user to quickly spot how many studies and quizzes the course contains. Each lesson is displayed in a row with details such as title, type, content count, and duration. The rows also include shortcuts for quickly changing the order of lessons and navigating to the Lesson Editor. The LessonManagement also provides buttons for adding new lessons, either a study or a quiz. There is also a button for publishing the course, making it available for regular users.

The development of these LMS admin functionalities, such as course and lesson management, was streamlined by reusing shared components from the SalesCue platform. For example, the ListView component is also used for displaying files and shares in the SalesCue.

The Course and Lesson editors were new components developed specifically for the LMS using React Hook Form, a library, that manages form state, validation, and submission, making it easy to create and maintain forms. The Course Editor component allows administrators to create a new course. The form includes fields for the course name, description, category, access rights, and a button to upload a thumbnail. React Hook Form manages the state and validation of these fields, ensuring that the form is submitted only when all required fields are correctly filled in. The Lesson editor, which is different for study and quiz lessons, is also managed by React Hook Form. For study lessons, the form includes fields for adding details and uploading files. For quizzes, the editor allows administrators to write questions and adjust settings such as the scoring and time limits. The same mechanism for handling forms is used elsewhere in SalesCue code base. Having prior experience with React Hook Form made it straightforward to implement the basic functionality for the Course and Lesson editors. However especially the development of the quiz editor was quite

complicated and required careful planning, iterative testing, and multiple revisions due to its complexity.

In addition to the course and lesson management views, there are two other main screens for admins that are necessary for a fully functioning LMS. The first is the LMS Settings screen, where admins can edit settings for the entire LMS. This includes the ability to set the title for the LMS (e.g., 'Learning Center' or 'Academy'), upload an image to be used as a banner on various LMS screens, and create and edit categories. This screen is planned for development and is expected to be part of the first version of the LMS.

The second important screen is the Analytics screen, where admins will have access to detailed data about all participants, their results, and their progress. This screen will also enable admins to enroll users in courses or send them notifications. The development of the Analytics screen is possibly going to be more complex, as it will require additional backend development and new endpoints. Advanced analytics features might be released in a subsequent version of the LMS.

The second category of views is designed for regular users. The main screens developed for this purpose are the Dashboard and the CourseCompletion screen, each catering to different aspects of the user's learning journey. The Dashboard is the primary interface where users can see all the courses available to them. Courses are grouped and displayed by category, making it easy for users to find the exact course they want. The topmost section of the Dashboard is reserved for courses that are currently in progress, allowing users to quickly resume from where they left off. The courses on the Dashboard are displayed using the CourseCard component, which visually resembles the Card component used for files and folders in the SalesCue app. Each CourseCard shows a thumbnail and some course details beneath, along with a progress circle, if applicable.

When a user selects a course from the Dashboard, they are taken to the CourseCompletion screen. This screen provides an outline of the course, displaying a short description of each lesson included in the course. There is also a summary using icons to show the total number of studies, files, and quizzes, as well as the overall duration of the course. On the right side, there is a sidebar that provides easy orientation and navigation, allowing users to quickly move between lessons.

When a study lesson is selected, the user sees the study's content and can open the files to inspect

them, whether they are PDFs, videos, or presentations. SalesCue's own technology for displaying media files, called Showroom, is utilized in the LMS to ensure a smooth viewing experience. Navigating to a quiz prompts the user to start the quiz, after which they will be presented with an assessment of their knowledge. However, at the time of writing this thesis, the quiz completion feature has not been implemented yet.

The `CourseCompletion` screen was developed specifically for the LMS. It is crucial for users to see their progress at all times and to receive immediate feedback from their actions, such as clicking on a file and seeing the progress bar update. The use of React hooks was particularly important for achieving this functionality. The `useParticipant` hook is directly connected to the service's `fetchParticipant` method, which fetches the participant data from the server. Whenever the participant's progress is updated, for example by opening a file, the `updateProgress` function not only updates the progress but also emits an 'update' event. The `useParticipant` hook, created using the previously mentioned `createDataSource` method (Figure 14), listens to these events and knows to fetch the updated participant data from the server upon receiving an 'update' event. This ensures that the participant data returned by the `useParticipant` hook is always up-to-date, and the progress is always displayed correctly.

Even though I know the SalesCue application's frontend development well and I was able to reuse some shared components, the process of developing LMS frontend still took a significant amount of time. This is due to the numerous views and actions available to users, as well as the many components that need to interact with each other. Additionally, the quiz completion feature has not yet been implemented. Developing this will be particularly interesting because it is a completely new feature that we don't have in SalesCue. Once the quiz feature is completed, the first version of the LMS will be nearly ready.



## 4 Conclusion

While I have enjoyed the development of the LMS, I also encountered certain challenges. One difficulty was adapting to changes in the database design or interface design after initial implementation. When these changes occurred, I had to carefully review the existing code and adjust it to align with the updated designs. This process was time-consuming and sometimes involved reworking complex features that were ultimately no longer needed. However, I understand that such changes are a normal part of the development process and are often unavoidable. Despite these challenges, the overall development process has been relatively smooth. Regular meetings with other team members allowed me to consult difficult points and receive valuable feedback.

One notable aspect of the development process is that creating new features often progresses easily at the beginning. However, the real challenges arise during the finalization phase. This phase requires careful consideration of all edge cases and addressing any unexpected issues. Bugs need to be identified and fixed, and the system must be tested to ensure it works as expected in all scenarios. The finalization phase is still ahead.

When working on the backend development, the fact that I was able to write the endpoints in TypeScript was very helpful, since I am well familiar with TypeScript well. Additionally, a colleague of mine handled the tricky server setups so I did not have to spend time with it. I enjoyed writing the endpoints because I could write tests to immediately test the methods in various scenarios. I also got a lot of valuable feedback from colleagues. I hope to have an opportunity to work with the backend development in the future too. The frontend development was made easier thanks to the well planned interface produced by the LMS product owner.

The main objective of this theses was to develop the minimal viable product of the Learning Management System. The goal was not fully reach, because the quiz functionality is still under the development at the moment of writing. However, there is not too much missing from the project. The release of the beta version for internal testing is planned in a couple of weeks so I hope that by then the first version of the LMS will be ready. The product is supposed to be available to Sales-Cue customers starting from January 2025, so there should be enough time for comprehensive testing and handling of issues.

The potential for the future development of the LMS is huge. There is already a plan for the phases 2 and 3 of the development and a long list of features it should include. The most significant ones are the course dependency, so that a certain course cannot be taken unless its prerequisites are completed, translation options for the whole course, more diverse options for the answers in a quiz, AI utilization in duration estimate and advanced analytics. I am excited to work on the LMS development in the future as well.

## 5 References

Bezhovski Z. & Poorani S. (2016). The evolution of e-learning and new trends. *International Journal of Educational Technology in Higher Education*, 18(1), 1–19. <https://doi.org/10.1186/s41239-021-00298-5>

Clark, R. C. & Mayer, R. E. (2011). *E-learning and the science of instruction*. Pfeiffer.

Ellis, R. (2009). *Field Guide to Learning Management Systems*. American Society for Training and Development. [https://home.csulb.edu/~arezaei/ETEC551/web/LMS\\_fieldguide\\_20091.pdf](https://home.csulb.edu/~arezaei/ETEC551/web/LMS_fieldguide_20091.pdf)

Islam, N. (2012). *Understanding e-learning system users' post-adoption usage behavior and its outcomes: a study of a learning management system*. [Doctoral thesis, University of Turku]. UtuPub Repository. <https://www.utupub.fi/handle/10024/77182>

Mitra, N. K. (2022). New Updates in Online Learning. In E. Babulak (Ed.), *New Updates in E-Learning*. IntechOpen. doi: 10.5772/intechopen.97987.

MongoDB (2024). *Introduction to MongoDB*. <https://www.mongodb.com/docs/manual/introduction/>

Morgan, G. (2003). *Faculty use of course management systems*. EDUCAUSE Center for Applied Research. <https://www.educause.edu/ir/library/pdf/ers0302/rs/ers0302w.pdf>

Nivasalo, M. (2023). *Full stack development in TypeScript with tRPC and React Native*. [Bachelor's thesis, Metropolia University of Applied Sciences] Theseus Repository. <https://urn.fi/URN:NBN:fi:amk-202304145267>.

React Native (2024). Core Components and Native Components. <https://reactnative.dev/docs/intro-react-native-components>

SalesCue, 2023. *What is sales enablement?* <https://www.showell.com/sales-enablement#what-is-sales-enablement>

Seppälä, J. (2017). *Development of Student-centred Language Learning Environment*. [Bachelor's thesis, Metropolia University of Applied Sciences]. Theseus Repository.

<https://urn.fi/URN:NBN:fi:amk-201705107380>.

Sørebø Ø. et al. (2009). The impact of e-learning on academic performance: A case study in teaching and learning. *Computers & Education*, 53(2), 389–399. doi:10.1016/j.compedu.2009.02.012.

tRPC (2024). *Docs*. <https://trpc.io/docs/v10>

Turner, D.B. (2015). *Utilizing LMS tools to help with student assessment in an online course*. *EAI Endorsed Transactions on e-Learning*, 2(6), 1–5. <https://doi.org/10.4108/el.2.6.e5>.

Zanjani, R. (2017). The important elements of LMS design that affect user engagement with e-learning tools within LMSs in the higher education sector. *Australasian Journal of Educational Technology*, 33(1). <https://doi.org/10.14742/ajet.2938>