Improving Teacher`s User Experience in a Virtual Learning Environment

Yannik Bauer
Master`s degree in Computer Science
Department of Computer Science, FCUP
2023

Orientador
José Paulo Leal, Assistant Professor, DCC – FCUP, CRACS – INESC TEC

Coorientador
Ricardo Queirós, Assistant Professor, uniMAD – ESMAD, CRACS – INESC TEC

MSC

U.PORTO

FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO

MSC

" I don't have problems; I just have more work to do! "

Seth Feroce

Acknowledgements

First and foremost, I sincerely thank my supervisor, José Paulo Leal, and co-supervisor, Ricardo Queirós, for their relentless support, guidance, and valuable insights throughout this research journey! Thank you for your time and all the interesting, funny, and enjoyable weekly reunions!

My appreciation also goes to the Department of Computer Science of the Faculty of Science of the University of Porto and the entire staff for providing a great environment for research, learning, and growth.

I am deeply grateful to INESC TEC and the ”FGPE Plus: Learning tools interoperability for gamified programming education” project supported by the European Union's Erasmus Plus program (agreement no. 2020-1-PL01-KA226-HE-095786) and financed by National Funds through the Portuguese funding agency, FCT - Fundação para a Ciência e a Tecnologia, within project LA/P/0063/2020, for financing me.

I extend my gratitude to all those who participated in evaluating the proposed systems and took the time to answer the questionnaires. Their feedback and insights were critical to the validity and reliability of this research and also provided interesting perspectives that enriched the depth and quality of this work.

I also deeply want to thank my colleagues and friends for the conversations, shared laughs, and for providing a listening ear during challenging times. Their company and moral support have made this journey a pleasure.

Without mentioning my family for their constant encouragement and support, no acknowledgment would be complete. So thank you for your support.

In sum, this journey would not have been possible without the collective effort, direct or indirect, of all those named and unnamed here. I am really grateful for the privilege of learning from each of you.

UNIVERSIDADE DO PORTO

Abstract

Faculdade de Ciências da Universidade do Porto
Departamento de Ciência de Computadores

MSc. Computer Science

Improving Teacher's User Experience in a Virtual Learning Environment

by Yannik BAUER

This thesis outlines the development of the teacher's User Interface (UI) and the back-end system for Agni, a web playground for learning JavaScript. The goal of this work is to deliver a good User Experience (UX) to teachers through an intuitive and user-friendly UI containing useful functionalities. Therefore, the focus shifts from the more commonly explored research on student experiences to the teacher. To realize this goal, the initial hypothesis was to use a headless Content Management System (CMS), Strapi. However, it was evaluated as unintuitive and time-consuming. Consequently, the development of a new UI became crucial, employing the headless CMS just for the Application Interface (API) and back-end data storage. The primary concept for the developed UI was to use the existing Agni student UI and make it editable. Also, several innovative features were integrated, including the integration of an external exercise repository, namely AuthorKit, the statistical analysis of student engagement, and content sequencing. Further, an experiment on generating programming exercises using GPT was conducted, and GPT was integrated into the system to assist teachers in the process of creating exercises. Finally, the evaluation of the developed UI revealed significant enhancements compared to the Strapi assessments, with an overall positive review. Evaluators primarily highlighted the intuitive and simplistic nature of the UI, thereby affirming the successful realization of the principal objective: to create a teacher's UI that is both intuitive and user-friendly.

UNIVERSIDADE DO PORTO

Resumo

Faculdade de Ciências da Universidade do Porto

Departamento de Ciência de Computadores

Mestrado em Ciência de Computadores

Melhorar a Experiência de Utilizacão do Professor num Ambiente de Aprendizagem Virtual

por Yannik BAUER

Esta tese descreve o desenvolvimento tanto da Interface de Utilizador (UI) do professor como do sistema de back-end para o Agni, uma plataforma web para aprender JavaScript. O objetivo principal deste trabalho é oferecer uma boa Experiência de Utilização (UX) aos professores através de uma UI intuitiva e amigável com funcionalidades práticas. Este foco diverge do estudo normal, que mais explora as experiências dos alunos em vez das dos professores. Para realizar o objetivo, a hipótese inicial foi a utilização de um headless Sistema de Gestão de Conteúdo (CMS), o Strapi. No entanto, Strapi foi avaliado como não intuitivo e desafiador. Por isso, o desenvolvimento de uma nova UI tornou-se crucial, utilizando o headless CMS apenas para Interface de Aplicação (API) e armazenamento de dados de back-end. O conceito principal para a UI desenvolvida foi utilizar a UI do aluno existente do Agni e torná-la editável. Alem disso, várias funcionalidades inovadoras foram integradas, incluindo a integração de um repositório de exercícios externo, nomeadamente o AuthorKit, a análise estatística do envolvimento dos alunos, e o sequenciamento dos conteúdos. Além disso, foi realizado um estudo sobre a geração de exercícios de programação usando GPT, o que depois foi integrado ao sistema para ajudar os professores na criação de exercícios. Finalmente, a avaliação da UI desenvolvida revelou melhorias significativas em comparação com a avaliação do Strapi, com um sentimento generalemte positivo. Os avaliadores destacaram a natureza intuitiva e simplicidade da UI, confirmando assim a realização bem-sucedida do principal objetivo: criar uma UI para professores que seja tanto intuitiva quanto amigável ao utilizador.

Contents

List of Figures

List of Tables

Acronyms

ACD Activity-Centered Design. 7

AI Artificial Intelligence. 3, 12, 13, 15, 24, 46

API Application Interface. 4, 19–21, 23, 24, 27, 28, 34, 36, 37, 41–44, 46–50, 55

BE back-end. 2, 11, 20, 21, 24, 27, 28, 34, 37, 42–44, 49, 50, 61

CMS Content Managment System. 3, 21, 24, 27, 28, 35, 37, 42, 43, 50

DOM Document Object Model. 22

FE front-end. 21, 23, 24, 27, 28, 37, 38, 42, 44, 49, 50, 61

FGPE Framework for Gamified Programming Education. 47

GDD Goal-Directed Design. 7

GPT Generative Pre-trained Transformer. 3, 4, 19, 23, 24, 28, 35, 37, 46–50, 56, 60–64

HCI Human-Computer Interaction. 6

ISO International Organization for Standardization. 6

LMS Learning Management System. 1, 3, 10–13, 16, 17, 34, 62, 63

LTI Learning Tools Interoperability. 63

MOOC Massive Open Online Course. 10

SASSI Subjective Assessment of Speech System Interfaces. 8, 52

SUS System Usability Scale. 8

UCD User-Centered Design. 7

UEQ User Experience Questionnaire. 8, 16, 17, 52

UI User Interface. 2–5, 7, 10, 11, 15, 19–24, 27, 29–32, 34, 35, 37–40, 42, 43, 45, 49–63, 77, 84, 85

UMUX Usability Metric for User Experience. 8

UX User Experience. 1, 2, 4–10, 16, 17, 25, 29, 37, 40, 46, 51, 52, 55, 60, 61, 63

VLE Virtual Learning Environment. 10, 61, 63

Chapter 1

Introduction

Programming has emerged as an essential skill in modern proficiency. According to LinkedIn's annual report for 2023, software development stands out as the top hard skill in demand, with related competencies such as SQL, Python, Java, and JavaScript also featured in the top 10 [1]. Nevertheless, learning programming is challenging for newcomers, as underscored by various studies [2–7]. According to Winslow [8], newcomers can grasp basic syntax and semantics, but the challenge often lies in combining these elements into functional programs. He further observes that it takes about a decade to turn a novice into an expert programmer. Numerous other studies [5, 9–12] emphasize practical exercises to learn to program. This is most efficient when complemented with quick and useful feedback. However, manually providing immediate feedback for each exercise is impossible for teachers. This is where automated assessment systems for programming exercises come into play, offering a learning environment for students and allowing teachers to focus on students who need more individualized support.

Numerous e-learning platforms have emerged over the years, aiming to enhance the educational experience. While several studies can be found primarily centered on students' experiences, concepts, and features to improve learning [9, 13–15], the teachers' User Experience (UX) often gets overlooked. A study from I. Maslov, S. Nikou, and P. Hansen points out that the teachers' UX can significantly influence that of the students [16]. When teachers find an e-learning system complex or not user-friendly, they may not leverage its full capabilities, which subsequently can impact the quality of students' learning experiences. A recent study revealed that only 52 percent of educators felt that a Learning Management System (LMS) simplifies instruction [17]. These studies highlight the need to focus not only on the student but also on measuring and enhancing the experience of teachers in e-learning platforms. This can be done with UX evaluations of existing systems or developing features to facilitate the teacher's use. Therefore, e-learning systems can be optimized to their full potential, creating an environment that benefits teachers and students.

1.1 Goal and Objectives

This thesis tries to enrich the current research space focused on the teachers UX and features in e-learning systems by delving into the design and development of the teacher's User Interface (UI) and back-end (BE) system for Agni. Agni is a web-based playground to learn JavaScript. At the start of the project, it consisted of an interface where students could learn through resources such as PDFs and videos and further apply their understanding with exercises that were assessed automatically. However, a dedicated teacher interface to create and manage the course contents and student enrolments with a BE system to store the data was missing. The central theme of this thesis, namely "Improving Teacher's User Experience in a Virtual Learning Environment", is to develop this missing part with a focus on refining the UX for teachers through an intuitive design and useful functionalities. Ultimately, evaluating the UX of the developed system becomes a key aspect of getting insights into external and teachers' perspectives.

The concrete objectives for this thesis are:

1. Good Usability and Intuitiveness: The interface designed for teachers must be intuitive and user-friendly. The system's evaluations emphasize its intuitiveness and the ease with which key tasks can be accomplished.

2. Facilitate Course Creation: The system should enable teachers to incorporate theoretical content such as PDFs or videos and support the creation of automated assessment tools such as quizzes and automatically-evaluated programming exercises.

3. Reuse of Contents: Reusing and easily adapting contents such as courses or exercises should be possible.

4. Content Sequencing: The platform should give teachers the flexibility to sequence course content. This means they should be able to determine the order and prerequisites of course resources for students, ensuring a structured learning journey and

the possibility of creating a game-like structure where students can progress after accomplishing a certain condition.

5. External Exercise Repositories: Teachers should be able to choose exercises from external repositories, enhancing the range of available content.

6. Assisted Exercise Generation: Given the emergence and growth of language-based Artificial Intelligence (AI) chatbots, such as Chat GPT, the integration of such a chatbot into the system to assist teachers in creating programming exercises was further proposed.

7. Integration with Learning Management Systems: The proposed system should integrate seamlessly with existing LMS platforms, facilitating easier adoption for teachers.

1.2 Approach

In pursuing the project's goals, Strapi, a headless Content Managment System (CMS), was initially selected for data storage and content creation and management with its built-in UI. However, a usability evaluation of Strapi showed it to be unintuitive and time-consuming to use. As a result, while Strapi was kept for content storage purposes, the design and development of a new interface for teachers was decided. This interface was implemented using Vue, ensuring its alignment with the existing student UI. A research paper, named "Can a Content Management System Provide a Good User Experience to Teachers", about the Strapi evaluation, initial design, and strategies of the developed UI was published at this stage [18]. Once the main parts of the new UI were developed, it underwent an evaluation with the same evaluators of the first evaluation to benchmark it against the original Strapi UI and five more professors. Compared to Strapi, the developed UI was evaluated significantly better. Also, the five professors evaluated the system overall positively despite some more critics. Overall, the system was seen as intuitive and simple. After that, and considering the received feedback, improvements, further refinements, and implementation of features such as the integration of Generative Pre-trained Transformer (GPT) and AuthorKit, an external exercise repository, were done.

1.3 Thesis Content

This thesis starts with exploring the existing research landscape in Chapter 2. The concept of UX is reviewed in Section 2.1, encompassing its definitions, diverse concepts, design guidelines, and various evaluation techniques. Subsequent analysis in Section 2.2 centers on existing e-learning systems, including system features, surveys, a study of popular e-learning systems, and a review of teachers' UX in e-learning systems. Concluding this chapter with a summary of the core findings in Section 2.3.

The next Chapter 3 presents key systems and metrics of this thesis. Starting with Agni in Section 3.1, followed by Strapi in Section 3.2 and VueJS in Section 3.3. Further, Section 3.4 describes the GPT API. The chapter ends with a summary in Section 3.5.

The following Chapter 4 reveals the system modeling with a top-level view of the concepts. Starting with a description of the system architecture in Section 4.1, which is followed by a presentation of the UI in Section 4.2. Next, a generalized data model is introduced in Section 4.3 followed by Section 4.4 about the Application Interface (API), including its permission rules. The chapter concludes with a summary in Section 4.5.

The subsequent Chapter 5 details the system's implementation. 5.1 explains the selection of Vue as the framework, the implementation of the course component, responsiveness, and integration of external content. Followed by 5.2, including Strapi's selection, how the data model was implemented, configurations to the API, and created unit tests. An experiment using GPT to generate programming exercises as well as its integration is described in Section 5.3. The main gatherings are summarised in Section 5.4.

The succeeding Chapter 6 details the conducted validations, starting with a presentation of the evaluation methodology in Section 6.1, which is succeeded by the assessment of the Strapi's UI in Section 6.2. The evaluation of the developed UI is subsequently presented in Section 6.3, including discussing its results both standalone and in comparison with Strapi's evaluation. The main conclusions of this section are compacted in the summary in Section 6.4.

In the conclusion in Chapter 7, the thesis structure and the accomplished and not accomplished goals are reviewed. Section 7.1 then presents the contributions and main findings of this project. Ending with the potential future work to improve the system in Section 7.2

Chapter 2

State of the Art

This chapter reviews the current state of the research on topics related to the thesis and establishes a foundation for the analysis and development undertaken in this thesis. Starting with an exploration into the domain of UX in Section 2.1, its diverse nature with different definitions and concepts is explained in Subsection 2.1.1. This is followed by presenting the different methodologies, principles, and strategies employed in a design process in Subsection 2.1.2. Next, the techniques for evaluating UX are explored in Subsection 2.1.3. Further in the chapter in Section 2.2, the focus is shifted toward e-learning systems. Their architecture, features, gamification, and criticism of automated assessment tools are explained in Subsection 2.2.1. A review of popular e-learning systems and their respective UI for teachers is presented in Subsection 2.2.2. A subsequent analysis of research on UX in existing e-learning systems is conducted in Subsection 2.2.3. Concluding this chapter with a summary of the key research and findings in Section 2.3.

2.1 User Experience

UX is a broad term with various discussions and debates regarding its definitions and concepts, both historically and currently. These discussions are detailed in Subsection 2.1.1. While understanding these concepts offers insights into the essence of UX, there are different methodologies and considerations for designing an optimal UX, as explored in Subsection 2.1.2. However, a development process isn't complete without evaluations to measure its effectiveness or end user's perspective and identify areas for improvement. Subsection 2.1.3 presents these evaluation methodologies.

2.1.1 Definitions, Concepts

The concept of UX is a broad field with various definitions proposed by different scholars. For example, the International Organization for Standardization (ISO) defined UX as "A person's perceptions and responses that result from the use or anticipated use of a product, system or service" in 9241-210. On the other hand, researchers such as Hassenzahl and Tractinsky interpret UX as the outcome of a user's internal state and the system's characteristics within a specific context of interaction [19]. Yet another definition by V. Roto articulates UX as feelings during and after interaction with a product, system, or object, influenced by various aspects such as user expectations, interaction conditions, and the system's ability to satisfy the user's current needs [20]. Given different authors' varied viewpoints and backgrounds, finding a unified definition of UX is challenging.

Diving deeper into UX, several theoretical models were presented to explore its different aspects. Hekkert, for example, divides product experience into Aesthetic Experience, Experience of Meaning, and Emotional Experience, referring to the interaction with a product, the symbolic meanings attached, and the emotions invoked during the use of the product [21]. In contrast, Hassenzahl [22] differentiates experiences into Pragmatic and Hedonic attributes, with the Pragmatic attributes relating to usefulness and usability and Hedonic attributes emphasizing the individual experiences such as stimulation, identification, and evocation. Expanding the discussion further, a study by N. Tractinsky, A.S. Katz, and D. Ikar highlights the importance of aesthetics in interfaces [23]. They asserted that aesthetics are intertwined with usability, significantly influencing Human-Computer Interaction (HCI) and UX during interaction. From another standpoint, P.W. Jordan [24] advocates for the inclusion of pleasure-based elements in the design, suggesting that designs should aim to evoke joy in users. Additionally, the idea of Co-Experience [25], which arises from the social aspect of human nature, can play a significant role in UX. It refers to the collective creation, elaboration, and evaluation of experiences, which can affect an individual's interaction with a product, both positively and negatively. UX is further considered an evolving concept, progressing through three main stages: Orientation, Incorporation, and Identification [26]. The Orientation phase involves initial interaction and familiarization with the product, the Incorporation phase reflects on the product's potential future value, and the Identification phase entails personal identification with the product in social contexts. Additionally, it's worth noting that while usability is an essential aspect of UX, they are not synonymous. Usability forms part of UX but does not capture the entire spectrum of experiences, including various emotional and contextual factors [27].

Despite the difficulty of defining UX due to its multifaceted nature, it is important to recognize the presented aspects. There is a consensus among authors that UX is a dynamic concept that extends beyond usability to include emotions, multiple usage contexts, and other intricate aspects, aiming to deliver a comprehensive and satisfying UX.

2.1.2 Design Strategies

Designing a product or a UI often involves considering consciously or subconsciously different approaches. Three commonly used are User-Centered Design (UCD), Goal-Directed Design (GDD), and Activity-Centered Design (ACD). UCD revolves around tailoring the product or design to suit the user's specific needs, considering their capabilities and limitations. GDD, on the other hand, focuses on achieving the user's ultimate goals through the design, defining how a product behaves based on the users' objectives. Lastly, ACD is centered around the activities or tasks the user undertakes, emphasizing the seamless integration of the product or design into the user's workflow. Several books or articles elaborate on the UCD process, providing detailed guidelines and insights [28, 29]. However, discussions continue about which approach is superior, with compelling arguments on all sides [30–32].

In addition to these design approaches, there are key concepts to effective design [28]. These concepts include Memory and Attention, Affect, Cognition, Perception, and Mental Models. The principle of Memory and Attention emphasizes that the UI should not be too information-heavy, as this could overload the user's cognitive capacity and reduce their ability to focus. The Affect refers to creating and responding to human emotions through design. A product or UI that can evoke positive emotions can result in a more satisfying and engaging UX. Cognition involves understanding how users think and process information. This understanding is crucial to developing designs that align with the user's cognitive abilities, improving usability and effectiveness. Perception relates to the design's interaction with the user's five senses. A successful design should consider how users perceive information visually, audibly, and even tangibly and utilize these senses to enhance the UX. Finally, Mental Models are the user's internal representations or beliefs about how a system works. These models guide user behavior and understanding. Designers should consider Mental Models to create intuitive and user-friendly interfaces.

These presented concepts can help designers develop a user-friendly, engaging, and effective product.

2.1.3 Evaluation

While carefully crafted, design choices might not always produce the anticipated impact on end users, as evidenced by various studies [33, 34]. As a result, evaluation stands as an essential phase in product or system development. Jakob Nielsen [35] highlighted four methods for assessing user interfaces: Automatic evaluation (computed), Empirical evaluation (user testing), Formal evaluation (utilizing models and formulas to measure usability), and Informal evaluation (rooted in the evaluator's expertise and general guidelines). Among these, empirical methods are the most popular. They include techniques such as Heuristic Evaluation, where usability experts analyze the system against established criteria; Cognitive Walkthroughs, in which evaluators simulate user tasks while interacting with designers about the product; and Pluralistic Walkthroughs, collaborative sessions involving both users and developers to explore the system [28, 35]. Though UX is a relatively new term, the concept of usability has existed for longer. Evaluating UX proves more challenging because it encompasses subjective elements such as pleasure and emotions, which are harder to quantify. On the other hand, usability, a crucial facet of a good UX, is more mature and objective, making its evaluation more straightforward. Consequently, many UX evaluation methodologies are built upon established usability techniques. In practice, usability evaluation methods often serve as tools for assessing the broader UX.

Several evaluation questionnaires are recognized in the UX field. One is the Usability Metric for User Experience (UMUX) crafted by Kraig Finstad [36]. It uses a four-item Likert scale to measure user perceptions of a system. This questionnaire took inspiration from the System Usability Scale (SUS), a tool for measuring usability [37]. Another questionnaire is the User Experience Questionnaire (UEQ) [38] or AttrakDiff [39]. A study by A. B. Kocaballi, L. Laranjo, and E. Coiera [40] categorized several UX evaluation questionnaires, including the AttrakDiff, SUS, and Subjective Assessment of Speech System Interfaces (SASSI) [41], into distinct UX domains such as hedonic, pragmatic, and engagement/flow. Their findings highlighted SASSI as the most comprehensive. Another review assessed the application of various UX questionnaires, particularly AttrakDiff, UEQ, and meCEU [42]. This study showed that the UEQ has gained significant popularity recently, with it being the predominant choice in research papers they examined. Their analysis also revealed that over 60% of the reviewed research employed multiple methods to assess UX.

Usability, with its more objective qualities, can be assessed by analyzing user logs or measuring the time users take to navigate a system or complete specific tasks. Further, it can be evaluated with questionnaires. Some of them were already presented. Another popular method is to evaluate Nielsen's 10 Usability Heuristics [43] for user interface design with a questionnaire.

1. Visibility of system status
2. Match between the system and the real world
3. User control and freedom
4. Consistency and standards
5. Error prevention
6. Recognition rather than recall
7. Flexibility and efficiency of use
8. Aesthetic and minimalist design
9. Helping users recognize/diagnose and recover from errors
10. Help and documentation

When using questionnaires as described, an important consideration is the number of evaluators required to produce reliable outcomes. Nielsen found that between 3 to 5 evaluators can identify 70-99% of usability issues [44]. He highlighted the effectiveness of several smaller evaluations rather than a singular large-scale review. Virzi's study reflected this sentiment, suggesting that 4 to 5 evaluators are sufficient to uncover roughly 80% of usability challenges [45]. In contrast, Hertzum and Jacobsen proposed a higher figure, suggesting that around 13 evaluators are necessary to detect the same percentage of problems [46]. Reviewing these and other studies, Hwang, Wonil, and Salvendy, Gavriel summarized the data to recommend approximately 10+-2 evaluators to discover 80% of usability issues [47].

2.2 E-learning Systems

E-learning systems have transformed education by introducing various digital platforms with new features for teaching and learning. This section first explores the architecture, features of e-learning systems, and criticism found on automated assessment tools in Subsection 2.2.1. This is followed by a review of well-known e-learning systems, emphasizing their teacher UI, design, and functionalities in Subsection 2.2.2. The section concludes by presenting research on the teachers UX within e-learning systems in Subsection 2.2.3.

To set the ground, there exist different types of e-learning systems, including LMS, Virtual Learning Environment (VLE), and Massive Open Online Course (MOOC). LMSs, such as Moodle, Blackboard, and Canvas, are platforms that handle the administration, documentation, tracking, reporting, and delivery of educational courses or training programs. LMSs focus on organizing and managing courses and tracking student progress. VLEs, such as Agni, offer digital spaces that mimic a classroom environment. They allow teachers and students to interact, communicate, view, submit assignments, and engage in other course-related activities. The emphasis is on course delivery and learner engagement. In contrast, MOOCs such as Coursera, edX, and Udacity offer open access to online courses. These courses allow students worldwide to engage in various subjects at their own pace. MOOCs support large numbers of students, far exceeding the enrollments of traditional classrooms or online courses within universities. They typically include recorded video lectures, readings, problem sets, and interactive user forums. While MOOCs significantly expands access to education, they also present challenges such as high dropout rates and the need for self-motivation and discipline.

2.2.1 Architecture and Features

Various e-learning platforms with automated assessment are presented in research, offering insides into their architectures and functionalities. One notable system is EduJudge [14]. EduJudge supports languages such as Pascal, Java, C, and C++. The platform is built on three pillars: the On-line Judge evaluation server, the crimsonHex exercise repository, and the EduJudge UI, a Moodle plugin. In the UI, educators can use the exercise repository, access Moodle's quiz capabilities, or organize competitions via QUESTOURnament, which allows bundling multiple exercises. This challenges students and lets them compete against their peers. Another system showcased by D. Muñoz de la Peña, F. Gómez-Estern, and S. Dormido offers similar functionalities but is tailored for C and MATLAB

[15]. Its teacher-centric UI allows educators to create and manage exercises and students. On the other hand, students have a dedicated interface to receive and work on exercises. Central to this setup is a server that stores data and evaluates student submissions. In the BE, students' codes are executed and composed with the instructor's evaluation code, allowing the teacher to write evaluation code for different kinds of exercises without being forced to input/output tests. Yet R. Queiros and J. Leal presented an approach for a standardized integration of various systems dedicated to one e-learning system with automated evaluation of programming exercises [48]. The architecture contains a repository for storing learning objects and exercises, an evaluation engine that assesses student solutions, an LMS to provide the exercises to students, an integrated development environment for coding, and a pivotal component that ensures seamless communication across all the modules. Lastly, Uncode is a web-based educational platform rooted in the INGInious [49]. The core objective is to deliver automated feedback on programming exercises to students. Given its foundation in INGInious, it can be integrated with LMSs through extensions. Various plugins have been developed to offer formative feedback, improving students' understanding of subjects and task execution efficacy. Uncode emphasizes formative and summative feedback to give students an explanation and clarification of their possible errors and optimize their learning strategy.

A review presented by J. Caiza and J. Del Alamo examined several automatic grading tools for programming [50]. This included mature tools such as CourseMaker, Webcat, and JAssess, as well as many other systems. A notable observation was that Java is predominantly supported across most systems. While some are standalone entities, others operate as extensions or plugins for LMS. This LMS integration benefits teachers because most universities are using LMS platforms. Commonly, the systems use various testing methods, including test cases, unit tests, and dynamic and static testing approaches. Another review also analyses automatic grading tools, highlighting systems such as CourseMaker, AutoGrader, and Mooshak [51]. The authors categorized their evaluation into three domains: programming exercises, users, and assessment results. Within each domain, they outlined levels of maturity characterized by specific features. For instance, in the programming exercises category, maturity was graded as Level 0: Manual configuration of exercises; Level 1: Capabilities for import/export of exercises; Level 2: Seamless integration with an exercise repository. Other categories had maturity levels, including functionalities such as user import/export or integration into an LMS system. On measuring these platforms against the maturity criteria, the review concluded that a big proportion, nearly half of these systems, did not attain a 50% maturity score. Among all the tools studied, Mooshak emerged as the frontrunner with a maturity rate of 83%.

However, despite using good e-learning platforms, learning programming is challenging, and maintaining student engagement is of essential importance. Gamification is a method to motivate students, but it can also motivate teachers. It involves including game-like elements in non-gaming environments, such as points, badges, leaderboards, and progress bars. In their study, Martinha Piteira, Carlos Costa, and Manuela Aparicio offer a holistic framework to shape gamified courses, spanning various dimensions: Target Audience, General Goals, Specific Objectives and Topics, Contents, Principles of Educational Design, Game Mechanics, Cognitive Absorption, Flow, Personality [52]. Another study underscores how gamification mechanisms, such as points, badges, and levels, have been successfully embedded into LMS such as Moodle [53]. The level mechanic is a pivotal gamification component, which necessitates sequencing exercises and course materials. This means setting criteria to control the order and manner in which students access and engage with particular exercises or content. Seqins is a robust framework designed to sequence course materials [54]. It is adaptable and can interface with any tool compatible with the IMS LTI specification. This compatibility ensures a seamless integration with LMSs. With Sequins, educational resources such as expositive materials (PDFs and videos) and evaluative components (assignments, tests, and exercises) can be sequenced. To further expand on the integration of systems with LMSs, J. P. Leal, R. Queirós, P. Ferreirinha, and J. Swacha provide a roadmap detailing integrating via LTI [55].

While automated assessment brings efficiency to the evaluation of programming exercises, there have been concerns about its binary nature, which often limits feedback to correct or incorrect verdicts. A student submitting correct code doesn't necessarily mean they truly understand the underlying logic [56]. Students often get caught in a cycle of making incremental changes, submitting repeatedly until they stumble upon the correct solution. Also, despite generating a correct code, many students struggle to articulate its logic, suggesting a possible disconnect between their learning process and the result [57, 58]. The rise of AI chatbots has added another layer to this challenge. Novice programmers can use AI tools to generate solutions. These chatbots are very accurate, especially with introductory problems [59]. This can diminish the learning of students. Addressing that concern, André L. Santos emphasizes the need for a more complete evaluation approach in his talk [60]. He advocates for posing questions during or after code submission to measure and verify students' understanding of their code. However, a challenge arises when attempting to automate this process. Creating questions that match a student's code can be complicated. An alternative strategy is providing feedback and hints during the resolution process. With such guidance, a struggling student might not be seeking answers externally.

2.2.2 System Review

The literature on e-learning systems predominantly focuses on student-centric features and the overarching design of the system. While this perspective is understandable, analyzing the teacher's side is also important. Therefore, a study was undertaken on various e-learning systems for learning programming. Given the diversity of such platforms and different strengths and functionalities, different kinds of e-learning systems were included. A central focus of the investigation was the content structure with expositives and evaluatives, the evaluation of exercises, how and if sequencing content is implemented, UI strategies, and other interesting features. Table 2.1 demonstrates the findings.

The analysis includes HackerRank and Coderbyte, systems for programming practice, and technical interviews. Given their popularity in the space, Udemy and CourseMaker were chosen for the MOOC representation. Mooshak2, a system dedicated to orchestrating programming contests and courses on the Web, was also considered. Additionally, Khan Academy was selected, offering pre-built courses and exercises where educators can enroll students and manage their progression. Although LMS might not have direct functionalities for programming, they offer an environment possible to extend. To represent this category, Moodle was chosen.

Starting with the content structure, platforms that support the creation of courses vary in their structural organization. Udemy and Moodle support a single-level structure, CourseMaker supports a two-level structure, and Mooshak2 a multilevel organization. This means that content such as exercises or materials (PDFs) can be structured into lessons, modules with lessons, or an infinite nested organization, as in Mooshak2.

		Hackerrank	Coderbyte	Udemy	CourseMaker	Mooshak 2	Moodle	Khan Academy
	Content Structure			1 Level	2 Level	Multilevel	1 Level	
Expositive	Pdf			✓	✓	✓	✓	
	Video			✓	✓	✓	✓	
	Article			✓	✓			
Exercises	Quiz	✓	✓	✓		✓	✓	
	Open Questions		✓	✓			✓	
	Programming Exercise	✓	✓	✓	✓	✓	Extensions	
	Exercise Rep.	✓	✓			Own	Own Questions	
	Import		Quiz				Quiz	
	External Ex rep.							
Evaluation	Test Class		✓	✓		✓		
	Input/Output tests	✓	✓		✓	✓		
	Test Types	Dynamic	Dynamic + Static	Dynamic + Static	Dynamic	Dynamic + Static		
Sequencing	Time					✓	✓	✓
	Exercise					✓	✓	
	For individual Students						✓	✓
UI	UI Strategy	form-filling	form-filling	form-filling	form-filling	form-filling	Editable Student UI	form-filling
	Preview		✓	✓	✓		✓	✓
	UI Helpers	Documentation	Documentation	Documentation + Tips		Tutorial	Documentation	Documentation
	Others	Cheating detection	Cheating detection	AI for generation				

TABLE 2.1: E-learning System Review

Further, nearly all platforms support diverse learning material formats, including PDF files and videos. Udemy and CourseMaker also allow the creation of custom HTML pages or articles to display content, giving the teacher more flexibility.

Regarding exercise creation, every platform, except for CourseMaker, supports multiple choice quizzes, which are naturally easy to evaluate automatically. Coderbyte, Udemy, and Moodle also offer capabilities for open-ended questions, although without any mechanism for automated assessment. While all platforms support the creation of programming exercises, Moodle necessitates additional system extensions. The presence of exercise repositories, however, shows some variation. Platforms such as HackerRank and Coderbyte offer repositories with pre-existing exercises and save instructors' custom-created exercises for future reuse. In contrast, Mooshak2 and Moodle only provide repositories for exercises or questions that educators created themselves. Import functionalities for exercises are limited, with only Moodle and Coderbyte allowing quiz imports. None of the platforms under consideration (despite indirectly Moodle) support integrating external repositories, which would increase the range of resources available to teachers.

Regarding the evaluation of programming exercises, each platform has its unique approach. Udemy exclusively relies on the implementation of a test class for unit testing.

In contrast, HackerRank and CourseMaker streamline the test creation process by allowing instructors to define input and output parameters. Mooshak2 and Coderbyte offer a more flexible approach, allowing the option to quickly set up tests using input/output parameters or create a test class for unit testing. Consequently, dynamic testing is always supported. On the other hand, static tests are only supported with systems that allow teachers to create a test class, which is not always the case.

The ability to sequence exercises, which defines when and how a student can engage with a specific resource (an aspect of gamification), is present in platforms such as Moodle, Mooshak2, and Khan Academy. Both Mooshak2 and Moodle offer sequencing based on time or the completion status of prior exercises for the entire class. Khan Academy provides only time-based sequencing, adjusting individual students or whole classes.

When examining the UI's design and its help mechanisms for teachers, the primary UI strategy across most platforms is a form-filling approach. This means teachers can input the information through fields arranged on the UI. Every presented platform employs this strategy except for Moodle. Moodle's interface for teachers mirrors the student's view but in an editable format, complemented with additional configuration and buttons. To bridge the potential disconnect where teachers might be uncertain about the student's view in a form-filling interface, platforms including Coderbyte, Udemy, CourseMaker, Moodle, and Khan Academy offer a preview feature. This allows teachers to visualize how their configurations and content will be presented to the students. As for support mechanisms to navigate the platform, most systems provide comprehensive documentation detailing usage. Udemy goes further, presenting teachers with tips on crafting engaging and interactive courses. Mooshak2 provides a tutorial that describes the UI's functionalities and buttons of the currently displayed screen.

The platforms discussed also introduce a range of other interesting features. For instance, both HackerRank and Coderbyte also incorporate anti-cheating mechanisms, including detecting ChatGPT use. Leveraging advancements in AI, Udemy has introduced an innovative feature wherein educators describe the desired programming exercise, and the system automatically creates it using a language-based AI chatbot. HackerRank is also venturing into AI integration, aiming to enhance the system.

In conclusion, while these e-learning platforms provide a range of features to assist educators in content creation and management, there's room for improvement. Specifically, they could benefit from better capabilities for reusing exercises or content, importing specific content, and integrating external libraries. These improvements would expand the platform's utility and give teachers more options. It's worth mentioning that although Hackerrank and Coderbyte offer robust evaluation features, they come at a higher cost compared to other platforms.

2.2.3 User Experience in E-learning

Many studies found in the research process explore the student's experience with online learning and how certain tools impact their learning [9, 13–15]. However, the teachers' experiences with these platforms are lacking exploration. As highlighted by I. Maslov, S. Nikou, and P. Hansen, when teachers find a platform challenging or not user-friendly, they might not make the most of its features [16]. This can, in turn, affect students' learning experiences. Therefore, it's important to consider teachers and students when evaluating online learning systems.

A recent study indicates mixed feelings among educators regarding LMSs [17]. While 40% found LMSs neither harder nor easier to use, just over 50% believed that LMSs simplifies instruction. The Blackboard system, in particular, was spotlighted for its UX challenges in another study [61]. Many educators described it as non-intuitive due to an overload of tools and links, leading to confusion. Meanwhile, A. Saleh, H. Abu Addous, I. Alannsari, and O. Enaizan analyzed Moodle's UX [62]. Using the UEQ questionnaire, they engaged students, teachers, and experts to evaluate the system. They didn't differentiate between student and teacher feedback, but the collective sentiment was positive, suggesting a good UX with Moodle. Lastly, another study provided an overview of the methods used to evaluate the UX in e-learning systems [63]. Their findings showed that over 50% of such research employed some form of inquiry, with questionnaires being the predominant method. Additionally, 33% incorporated evaluations with hands-on tests, and 27% leaned towards inspection-based evaluations.

Reaching a consensus on the current UX for teachers in e-learning systems is challenging. One primary reason is the lack of studies that specifically evaluate UX from the teachers' perspective. Often, results from teachers and students are aggregated, with fewer teachers included in the study. This makes a clear understanding of teachers' experiences difficult. Also, the research on LMSs presents mixed findings. For instance, one study presents a positive assessment [62], while another leans towards a neutral stance [17], noting many educators don't necessarily find LMS tools simplifying instruction. Meanwhile, another study highlights the unsatisfactory experience with Blackboard [61]. Given these disparities, there's a need for more research focused on teachers. Enhancing their UX can further improve e-learning systems, benefiting both teachers and students.

2.3 Summary

This chapter described the multifaceted domain of UX. At its core, UX is a dynamic concept, encompassing various elements from emotions and usability to varying usage contexts. When designing for UX, ensuring that the system resonates with users' pre-existing mental models is important [28]. Also, evaluating a developed interface is essential, as design choices don't always produce the expected outcomes for the end user [33, 34]. Plenty of evaluation methodologies are available, with popular ones being questionnaires such as AttrakDiff and UEQ, as well as usability evaluations based on Jakob Nielsen's 10 usability heuristics [38, 39, 43]. Regarding the ideal number of evaluators, there exist different studies ranging from 3-5 up to 13 evaluators to identify most usability issues [44–47].

The section about e-learning systems explores their architecture and integration with different systems. Gamification is an interesting concept to motivate students and teachers for better learning and teaching. However, criticism of the binary nature of automated assessment tools is presented [56]. There was a disconnect between a correct solution and the appropriate learning or understanding of students' codes found [57, 58]. To solve this, asking students questions about their code after having resolved exercises was proposed as a possible solution [60]. Subsequently, a detailed review of well-known e-learning systems reveals untapped potential in reusing exercises, importing content, and integrating external libraries. The research of teachers' UX within e-learning systems is limited. While the existing studies span a spectrum from negative to positive outcomes, drawing real conclusions remains challenging [17, 61, 62]. Nevertheless, it's important to address the teacher's perspectives more, as the UX for teachers can significantly influence students' UX and learning outcomes [16].

Having reviewed the research associated with the themes of this thesis, the next Chapter 3 will shift the focus to explain the systems underlying this thesis.

Chapter 3

Background

This chapter introduces the key systems used in this thesis. Each system will be explored in detail, highlighting its intended use, features, strengths, and potential drawbacks to provide an understanding of their roles within the broader context of this thesis. Starting with Agni, a platform for learning JavaScript in Section 3.1. This is followed by Strapi, a headless CMS in Section 3.2 and Vue, a framework for UI development Section 3.3. After that, Section 3.4 explains the API and the language-based Generative Pre-trained Transformer (GPT) model. The chapter concludes with a summary in Section 3.5.

3.1 Agni

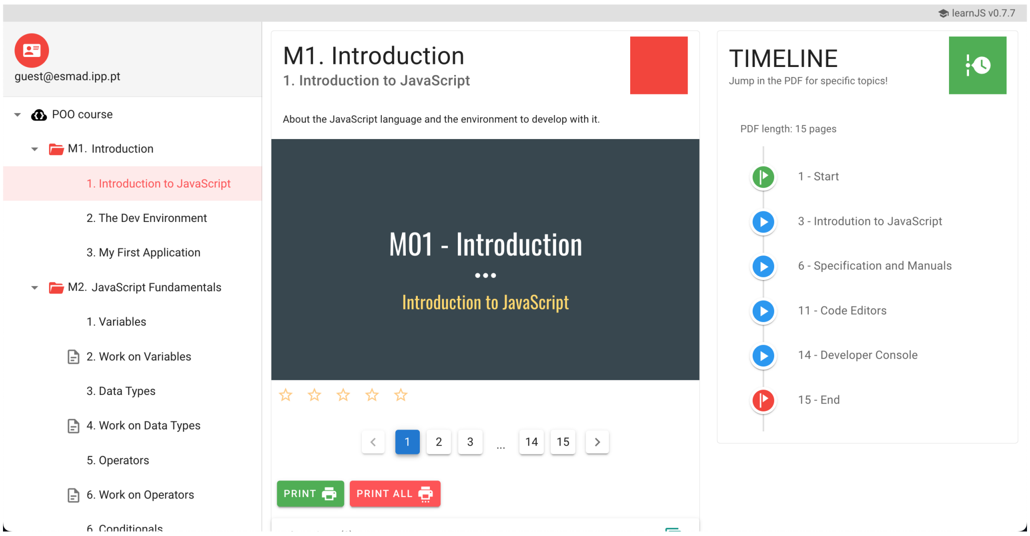

FIGURE 3.1: Agni previous Student UI

Agni is a Web playground for learning and practicing the JavaScript language [64]. The system was developed using Vue 2 and consists of two main components: the Editor and the Evaluator.

The Editor component UI, showcased in Figure 3.1, is an interactive learning space for students. Designed for simplicity and ease of use, this interface facilitates learning through materials such as PDF files and videos. It enables students to engage in practice exercises, including quizzes and programming exercises. The data structure is illustrated in Figure 3.2. Within this structure, a course has modules that contain resources. These resources can be expositive materials or sheets that are a collection of exercises. Furthermore, exercises can be categorized into quizzes with questions or programming exercises with automated evaluation tests.

The Evaluator component assesses the students' code based on static and dynamic analysis parameters. Dynamic analysis involves functional tests run on test cases to verify if the code is correct, efficient, and free from memory leaks. Meanwhile, static analysis identifies syntax errors, unused variables, and potential bugs due to implicit type conversion. For static analysis, Agni employs JSHint, a static code analysis tool. Importantly, all evaluations are performed client-side, ensuring quick responses.

Its 0.7.7 version only has an interface for students, lacking functionalities for creating new courses, modules, or resources. Only a single course was accessible, stored within configuration files. This thesis addresses those gaps by introducing a teacher's interface, allowing for course and content creation and management. This addition, complemented by an Application Interface (API) and BE system for data storage, aims to complete the Agni platform.

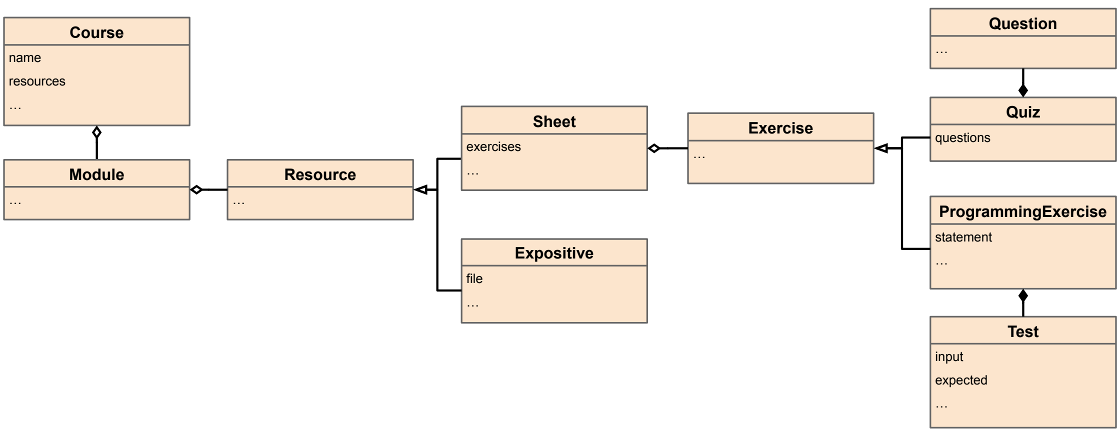

FIGURE 3.2: Agni previous Data Model

3.2 Strapi

Strapi is a headless CMS. While traditional CMS software allows users to create, manage, and modify digital content on websites without deep coding expertise. It typically connects the FE, where content is displayed, with the BE, where content is managed and stored. This combination can lack flexibility, especially when updating the FE or using the content across various platforms. In contrast, as suggested by its name, a headless CMS focuses solely on the BE. The FE, or the head, is managed independently. The BE usually incorporates an API to interface with the FE. This separation allows changes to the FE without disrupting the core content. Moreover, it facilitates content integration across multiple platforms via the API.

Built on NodeJS, Strapi is an open-source platform with a user-friendly interface, as shown in Figure 3.3. This interface facilitates the construction of content structures and content creation. Users can define fields for various content types, including Collection types, Single types, and Components. Collection and Single types represent content structures with an API endpoint. These can be created, edited, and managed independently. On the other hand, Components are reusable structures that can be used within a Collection or a Single type. The REST API supports the common CRUD operations, create, read, update, and delete, with filtering through various parameters. To further enhance its capabilities, Strapi allows for customizing API responses and actions through hooks. These include Collection hooks, which modify response behavior, and Webhooks, which can trigger subsequent requests based on predefined criteria. However, such configurations necessitate manual coding. Moreover, Strapi provides a range of plugins. Among these is a user permission or an email plugin. Its open-source nature encourages the community to develop and integrate additional features, expanding its capabilities. Strapi defaults to SQLite for data access, allowing switching to more scalable databases such as PostgreSQL, MySQL, or MariaSQL. Additionally, Strapi facilitates the integration of unit tests with Jest to ensure system reliability.

Within this project's scope, Strapi was employed for BE operations and data storage. Initially, its UI for content creation was considered a possibility for teachers to manage content. Yet, due to limited configurability and suboptimal results from a usability evaluation, detailed in Chapter 6, its role was narrowed to data storage and communication via the API.

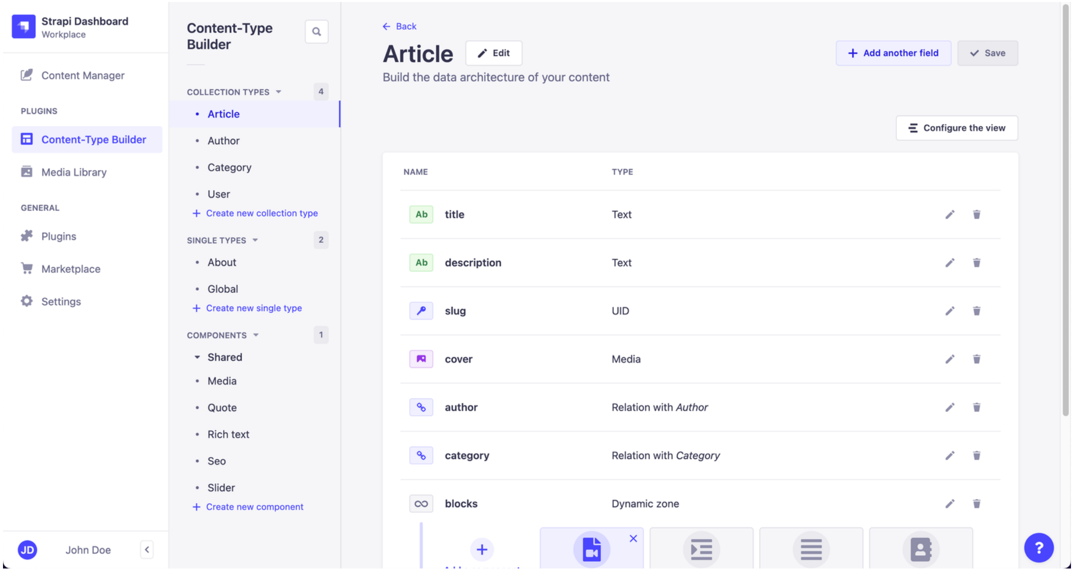

FIGURE 3.3: Strapi's Content-Type Builder Interface

3.3 Vue

Vue is a progressive JavaScript framework to develop UIs and single-page applications. It was created by Evan You, a former Google engineer, in 2014, and since then, it has gained significant popularity in the development community.

Vue is said to have a gentle learning curve, and developers can quickly build simple applications. Its simple structure and detailed documentation make it a great choice for beginners and experienced developers. Vue is component-based, similar to React and Angular. This means developers can build small, self-contained, reusable components and combine them to build complex UIs. Every component has its state, markup, and style. This architecture makes it easier to test and maintain large applications. One of Vue's unique features is its directive system, which allows developers to apply special behaviors to the rendered Document Object Model (DOM). For example, the v-if directive can conditionally render elements, the v-for directive can render a list of items based on an array, and the v-model directive can create two-way data bindings on form inputs. Vue uses a virtual DOM, which makes it extremely efficient. When a Vue application's state changes, instead of immediately updating the real DOM, Vue creates a copy of the DOM (known as the virtual DOM), applies the changes there, and then compares the virtual DOM to the actual DOM. Only the differences are then updated in the real DOM. This process significantly improves performance.

However, Vue does not support all needed functionalities inherently. Therefore, various libraries have been developed to extend its capabilities. The most common ones are Vuex, Vue Router, and Vuetify. Vuex is a state management library for Vue. It acts as a centralized store for all components in an application, ensuring a consistent management of state data across them. Vuex facilitates the organization of states, actions, mutations, and getters, making the development process more structured. Vue Router is the official routing library for Vue. It allows defining routes and navigation through the application without requiring a full page reload. Vuetify, on the other hand, is a material design framework providing many pre-built UI components. This framework facilitates the construction of the UI, allowing developers to create responsive, mobile-first projects on the web without investing too much time in styling and design. It offers components such as navigation bars, footers, icons, and buttons that align with Material Design principles, providing a consistent and modern aesthetic across applications.

In conclusion, Vue's simplicity, flexibility, and performance make it a powerful tool for web development. Its gentle learning curve makes it accessible to beginners, while its features and flexibility make it useful for developing complex applications. This thesis chose Vue as the FE framework for developing the teachers UI.

3.4 OpenAI GPT Application Interface

The OpenAI API, leveraging the GPT architecture, is a tool that facilitates the generation of contextually and human-like text. This API is trained to share information, respond to questions, create content, and propose suggestions, making it a valuable asset for various research domains. It can analyze and understand language constructs, allowing it to engage in dialogues.

Using the GPT API includes costs based on the chosen model and the volume of information sent, quantified in tokens. Additionally, there are rate limits for the tokens and the number of requests possible to send. Typically, three requests in a minute and 200 requests per day are allowed. While requesting an increase in these limits is possible, a good argumentation with data supporting the need is required for approval.

Despite its advanced capabilities, GPT has its limitations. Due to its knowledge cutoff in January 2022, it lacks real-time or up-to-date information. It cannot learn or store personal user data, thereby being unable to offer personalized learning experiences or recall previous interactions. The inability to perceive emotions or holistically comprehend human experiences restricts GPT from providing responses that require deep emotional understanding or empathy.

Several other AI models coexist in the technological landscape, emphasizing varied functionalities. Another language-based model, similar to GPT, is Bard, from Google. In addition to text-based models, AI has made significant progress in the image and audio generation fields. Models such as DALL·E by OpenAI can create images from textual descriptions. Similarly, models such as Jukebox, also from OpenAI, have ventured into the audio domain, synthesizing music and demonstrating the potential of AI in creating it.

GPT and its contemporaries represent the ongoing evolution of AI, each contributing to the exploration and application of artificial intelligence. While GPT excels in natural language processing and generation, other models bring innovations in visual and auditory domains, showcasing AI's versatility and expansive potential. However, it's important to acknowledge these technologies' limitations and ethical considerations to ensure responsible use.

3.5 Summary

This chapter presented the key systems underlying this thesis. Starting with Agni, a web playground for learning JavaScript, was presented with its user-friendly UI for students. It supports quizzes and programming exercises that can be assessed with dynamic and static tests. This thesis completes Agni with a teachers UI to create and edit courses and student enrollments and a BE system to save the contents.

Further, Strapi, a headless CMS, is presented. Its user-friendly UI, which facilitates the creation of content structures and offers options for customizing API responses using hooks and plugins, such as user permissions or email services, makes it a good choice for the BE system for Agni.

Vue is a FE JavaScript framework used for developing the Student UI and the Teachers UI of Agni. It is said to have a gentle learning curve and has a component-based architecture. Vue employs a virtual DOM, optimizing efficiency in data handling. To further enhance its functionalities, the main libraries are Vuex for state management, Vue Router for defining routes, and Vuetify, a material design framework.

The OpenAI API provides the use and interaction with GPT, an AI language-based text generator. Its use is limited to a few requests and tokens per minute. It can respond to questions, create content, and propose suggestions, making it a powerful tool. However, this comes with the limitations of a knowledge cut-off in January 2022. Despite this limitation, it was used in this thesis to assist teachers in creating programming exercises.

After presenting the literature concerning UX and e-learning systems in the previous Chapter 2 and outlining the foundational systems in this chapter, the basis is set to explain the proposed system, starting with the system modeling and its concepts in Chapter 4.

Chapter 4

System Modelling

Following a presentation of the existing literature and systems underlying this thesis, this chapter will dive into the concepts and design aspects of the proposed system. The developed teacher's UI and a BE system, which will be presented, complete the Agni system that previously only included a student UI. The exploration begins with the System Architecture in Section 4.1, providing an overview and detailing how each component interacts and functions as part of the system. Subsequently, the UI in Section 4.2 is detailed. The section explains the design concepts and interactive elements of the proposed teacher's UI. After that, the data model is presented in Section 4.3, including how the data is organized to provide the wanted functionalities. Finally, attention is given to the API in Section 4.4, where its communication between the BE and FE is presented. The chapter ends with a summary in Section 4.5, recapping the key points.

4.1 System Architecture

This section presents the proposed system's architecture with an overview of the key components and their interaction. It sets the stage for a deeper understanding of specific elements such as the UI in Section 4.2, Data Model in Section 4.3, and API as central role for communication between the FE and BE in Section 4.4.

The proposed system's architecture, as displayed in Figure 4.1, is primarily built on a headless CMS concept, where normally the headless CMS servers for the content storage, management and creation. The FE is separate for content display. However, the presented architecture differs from this by having the FE with functions of managing, creating, and

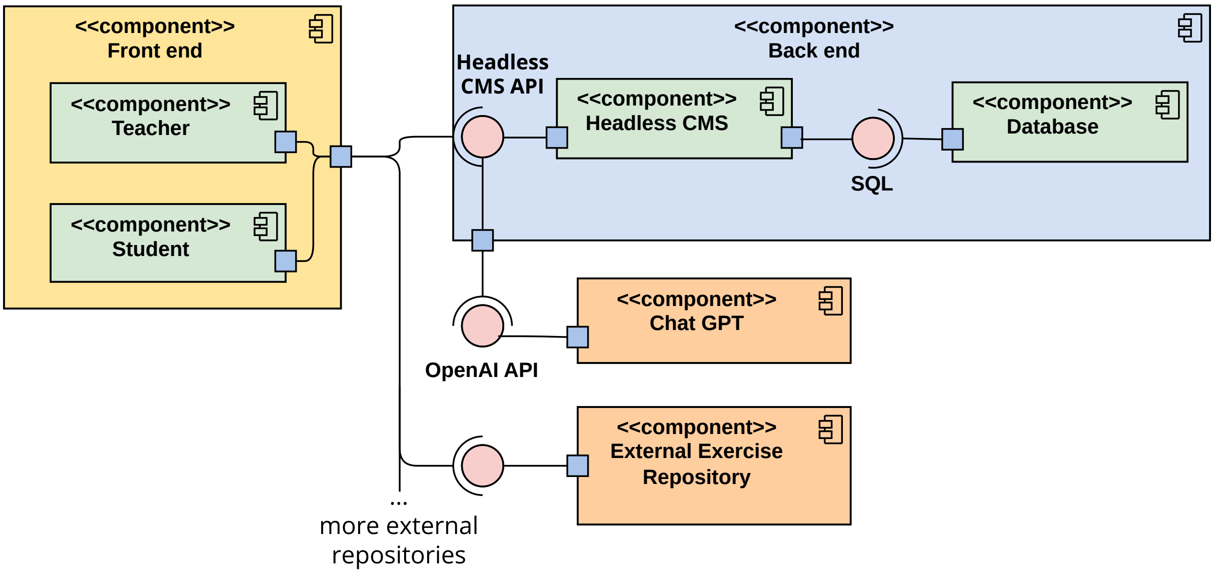

FIGURE 4.1: Generic System Architecture Diagram

displaying content. The system's FE can be divided into two parts, one designed for students and another for teachers. The student-oriented interface displays course content and supplies students with practice exercises. Contrarily, the teacher-oriented interface serves to manage and create course content, thus expanding the conventional role of the FE beyond just content display. The BE, akin to a headless CMS, incorporates the CMS and a database for content storage. It communicates with the FE through its API, headless CMS API in the displayed figure. Additionally, in line with the project objectives, the FE can interface with external services, allowing the integration of exercises from external repositories. The use and integration of GPT have costs and credentials associated. Therefore, a direct interaction between the FE and OpenAI API is avoided due to a higher risk of exposing these credentials to attackers. To interact with GPT, the FE sends a request to the BE API, which then sends a request to the OpenAI API.

This architecture leverages the benefits of a headless CMS, specifically decoupling the FE, and allows for easy updates and modifications to the FE without necessitating a system-wide alteration. Furthermore, the BE's configuration with its API and data storage enable potential data usage by other services, mirroring how the FE interacts with external services.

4.2 User Interface

This section explores the design principles and patterns in developing the teacher's UI. The UI is a critical bridge between teachers and the system, significantly shaping the UX. A well-structured UI can simplify complex tasks, reduce cognitive burden, and boost user satisfaction. In the proposed system, significant effort has been dedicated to creating a UI that enhances usability and intuitiveness.

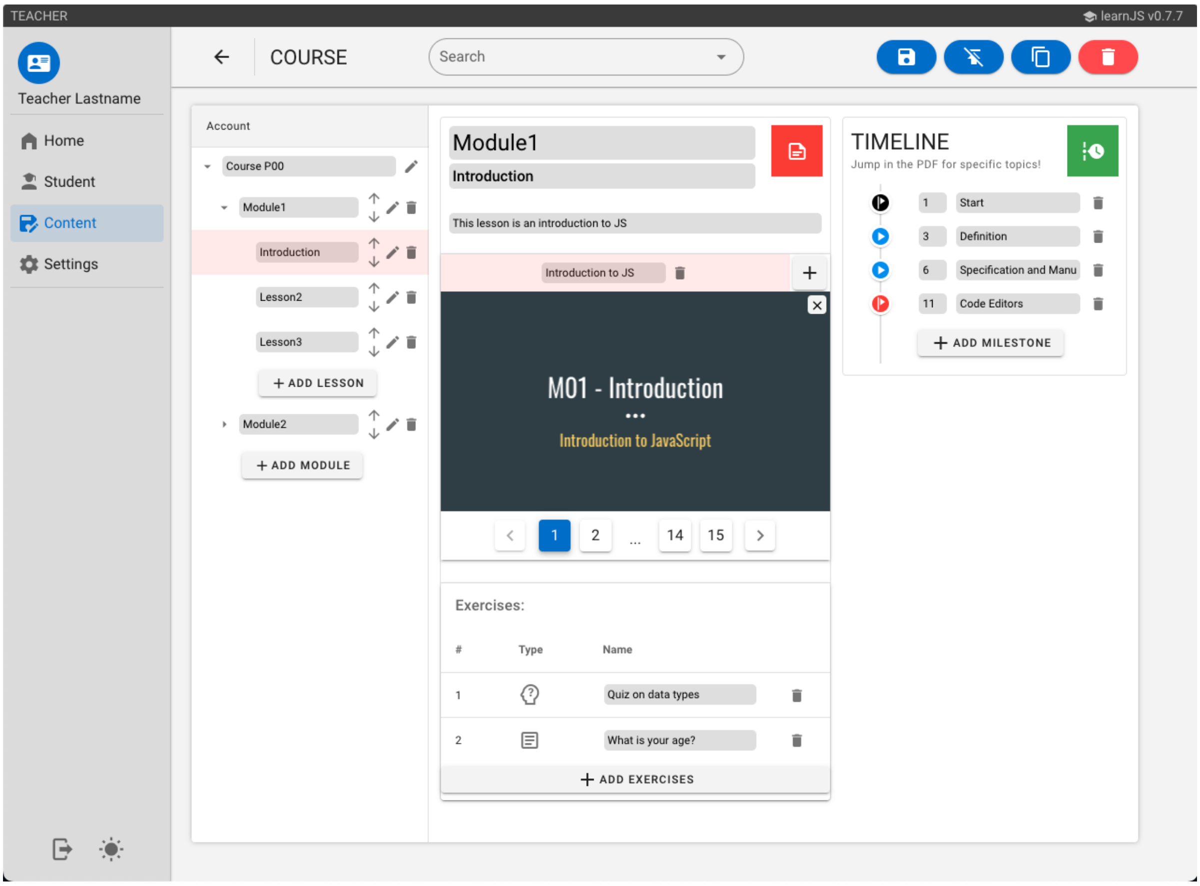

FIGURE 4.2: Interface to create or edit a Course

The designed UI, displayed in Figure 4.2, follows a three-part layout. It includes a menu on the left for main navigation, a top bar for additional tasks and information, and a large central area for most activities. This design is common in web applications, therefore trying to fit users' mental models, helping them feel familiar and navigate the system more easily based on their previous experiences with other websites. The design uses a clean, simple style with blue, gray, and white colors. Red and green indicate errors or deletions and successful actions, respectively. A focus was also placed on icon use, promoting quick recognition over textual explanations. Further, it was tried not to overload the user with functionalities and text so that key buttons can quickly be recognized and the overall structure is rapidly understood.

The system's UI has two primary functionalities:

- Student managing: The creation of occurrences with classes and students and reviewing their progress.
- Content managing: The creation and adaptation of courses, expositives, evaluatives, and questions.

For managing students, the UI distinguishes past, current, and future occurrences wherein students enroll in classes. The current occurrences are made more visible for rapid and easy access. Teachers can create, delete, and import students and classes, associate them with a course, and visualize the course layout in the defined time interval. As explained in more detail in the data model Section 4.3, a course has modules and lessons that can be sequenced based on time or exercise completeness conditions. Therefore, for teachers to easily understand the layout of their selected course, it can be displayed in two ways:

1. Using a calendar to display when a student can assess a certain module or lesson. This is appropriate for courses that are mostly sequenced by time-based conditions.
2. Visualizing all the modules, lessons, expositives, and evaluatives in a tree view, showcasing currently available resources with different colors.

Furthermore, statistics can be displayed to better understand the student's progress in a course. Three primary metrics were declared: percentage of correct exercises, engagement, and overall performance.

1. Percentage of Correct Exercises:
   To calculate the percentage of correct exercises done by a student, the number of exercises the student has completed with a grade of 100% is considered. This number is divided by the total number of exercises available in the course.

   - c: The number of exercises with a grade of 100% (correctly solved).
   - t: The total number of accessible exercises.

The equation for the percentage of correct exercises (P) is given by:

$$P = \frac{c}{t} \times 100\% \tag{4.1}$$

2. Engagement:

   Engagement is based on the student's progress in the recent two lessons. For this, the sum of all grades of the exercises from these two lessons is divided by the total number of exercises in these lessons. The grade of an exercise is on a scale from 0 to 100, with 100 being the perfect score. ,

   - g: The sum of grades of all exercises from the last two lessons.
   - n: The number of exercises in these last two lessons.

   The equation for engagement (E) is:

$$E = \frac{g}{n} \tag{4.2}$$

3. Performance:

   The performance metric averages the percentage of correct exercises and engagement. Therefore, the two mentioned metrics are summed and then divided by 2 to get the performance.

   The equation for performance (F) is:

$$F = \frac{P + E}{2} \tag{4.3}$$

Further, the exercises and the percentage of students that resolved them correctly can be seen, as well as the submitted solutions of each student.

For managing pedagogical content, the UI differentiates by course, expositive, evaluative, and question, displayed in tables in the content dashboard. The interface for creating or editing mirrors the student-facing UI, as shown in Figure 3.1 and the teacher's UI in Figure 4.2. For the teacher's version, text fields are converted into input fields, with additional interactive elements such as buttons or icons for adding or deleting modules, lessons, or exercises. This approach lets teachers see a live preview of how modifications will appear to the student, improving intuitiveness and ease of use. The same principle applies when creating individual expositives, evaluatives, or questions. Teachers also can view, reuse, and adapt content from other authors.

Additional UI features have been introduced to assist teachers. A search function in the top bar allows teachers to easily locate various elements such as exercises, students, modules, and course goals, which then directly access the specific interfaces, whether a course from a module or an occurrence from a student. Both a light and dark mode have been incorporated for individual preferences. Furthermore, teachers can upload a profile picture to personalize their profiles.

4.3 Data Model

This section explains a generalized data model to understand how the data is structured and the key classes to support the wanted functionalities.

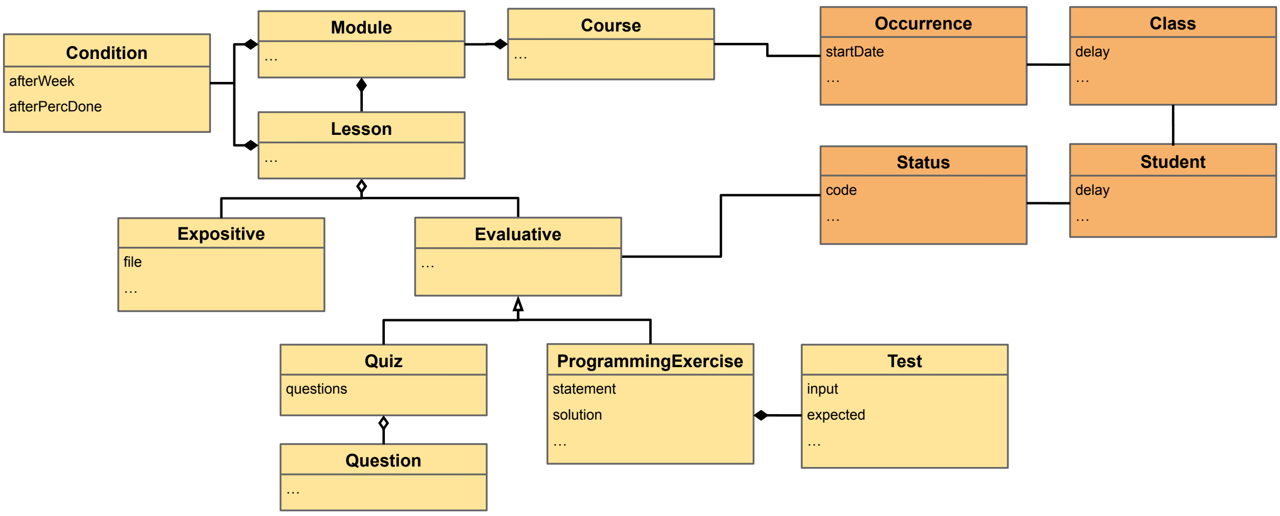

FIGURE 4.3: Generalized Data Model

In Section 3.1 was presented the previous data model of Agni in Figure 3.2. The course, which was only one at the beginning, was stored in configuration files. Some alterations and additions to this model were made to complete and refine its structure. A generalization of the new data model with the key classes and only a view fields is displayed in Figure 4.3. It can be divided into two parts: one for managing the students, in orange, which teachers can access in the student management navigation, and the other for managing courses and their contents, in yellow, which teachers can access in the respective content management navigation.

Starting with the content management part, which consists of a `Course`, that is composed of `Modules`, which contain `Lessons`. Previously, rather than `Lessons`, Agni directly had `Resources`, either `Expositions` or `Exercise sheets`. Recognizing the structure of school or university courses, which provide both instructional content and hands-on learning opportunities within most lessons. The decision was made to transition to a lesson-based format. Within this structure, each `Lesson` can have multiple expository materials, PDFs, or videos with exercises for students to test their understanding in programming exercises or quizzes. The programming exercises further can have `Tests` with `input`, `expected output`, `type` (log, expression, metric, function), and `subtype` (error for expression, occurrences, and lines for metric) parameters. This gives teachers a quick way to create different types of tests, including dynamic and static tests, without having to write full code for unit tests. To sequence, which was one of the declared objectives, `Modules` and `Lessons` contain conditions with fields named `afterWeek` to specify after which week the student can work on a `Module` or `Lesson`, and `afterPercDone` to define a percentage of completed exercises after which the student can progress.

The `afterWeek` condition works on the lessons and modules. At the module level, `afterWeek` represents the specific week after a module becomes available. This ensures learners follow a structured timeline, preventing them from advancing too rapidly. Once this time-based threshold is met, the lessons within that module become candidates for access. However, individual lessons can also have their own `afterWeek"` conditions. This offers a more refined sequence for lesson availability. For example, a module might become available in the third week, but a specific lesson within it might be set to the fifth week. The `afterPercDone` condition focuses on a learner's progression by evaluating the percentage of correct exercises, meaning with a grade of 100. At the module level, the availability of a new module requires learners to finish a predetermined percentage of exercises from the preceding module. For instance, if a module has an `afterPercDone` set at 80%, it implies that a learner must have correctly submitted 80% of the exercises from the prior module to unlock the subsequent one. Lessons also can have a `afterPercDone` condition. To unlock a new lesson, a student must complete the defined percentage of exercises from the lesson before it. When dealing with the first lesson in a module, this condition references the completion rate of exercises from the final lesson of the preceding module.

`Occurrences`, `Classes`, `Students`, and `Statuses` form the structure for managing students, which didn't exist in the previous version of Agni. `Occurrences` are linked to a `Course` and have a start date, determining when students can start accessing the `Course`. The `afterWeek` condition for `Modules` and `Lessons` is based on this start date. `Classes` and `Students` have fields for declaring delays that may occur during the year. The delays work summative, meaning that in a case of associating a delay of 2 weeks to a class, all students in that class with no delay associated will have a delay of 2 weeks. Students with a delay of 1 week associated will then have a delay of 3 weeks. Further, the grades and solutions to exercises done by the student are saved in the `Status`.

With this data model, a structured course can be created and sequenced. Further, different learning materials and the two main automated evaluation exercises, quizzes, and programming exercises are supported. Also, organizing classes and students in a similar structure to universities will make it easy to import and integrate the systems with university or LMSs. As a consequence of the alterations in the data model, the student UI underwent modifications, as displayed in slightly alternated UI in Appendix A and previous UI in Figure 3.1.

4.4 Application Interface

Having detailed the UI, in Section 4.2, and the data model, in Section 4.3, stored in the BE. This section presents the API, the component for communication between the BE and the UI.

Therefore, a API is required that supports CRUD (Create, Read, Update, and Delete) operations. These operations should apply to the key classes, namely Course, Expositives, Evaluatives, Occurrence, Students, and Statuses, for easy reuse, individual alterations, and visualizations. Additionally, query parameters such as pagination, filtering, search, etc., need to be incorporated into the API, facilitating the retrieval of specific content variations. This ensures a more organized and efficient content search. Moreover, a complete search feature enabling a search across all content categories, including courses, materials, students, and occurrences, is necessary.

The system's BE has both students and teachers who will interact with the API to get and post data. Therefore, the definition of permissions for interaction is a key aspect. Two main roles are defined: `Student` and `Teacher`, as shown in Figure 4.4. Further, the role of `Teacher` can be divided into `Author` and `Viewer`. Students are provided with access

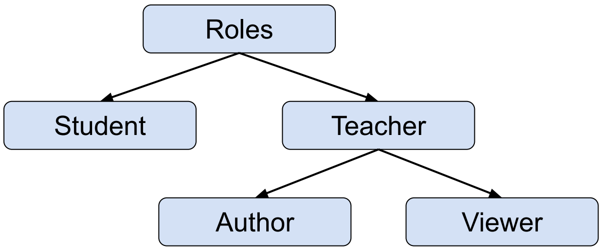

FIGURE 4.4: User Permission Roles

to their assigned courses. The content visibility is dependent on their progress and the course timeline. Additionally, students can update their exercise progress, necessitating PUT requests to status. The role of `Teacher`, as explained, can be further defined as `Author` or `Viewer`. Regardless of their categorization, all teachers can create all classes visible in the data model. However, their update privileges are limited to the objects they have created. They can access all content classes, which include their exercises and expositives, as well as those published by their peers. For occurrences, classes, students, and statuses, teachers can only retrieve those they have personally created. This approach ensures student privacy and enables teachers to view and clone course content or exercises created by other teachers.

4.5 Summary

This chapter offered a bird's-eye view of the developed system, introducing key designs and foundational concepts. The architecture is similar to a headless CMS concept. However, it is complemented by a custom UI for content creation and management. Also, external services such as external exercise repositories and chat GPT are integrated.

The UI section showcased a common three-part structure consisting of a primary menu on the left, an app bar for secondary tasks, and a main area for content display and interaction. A central concept was transforming the student UI of Agni into an editable format, enabling a user-friendly and intuitive approach to course creation. Also, the intention is not to overload the interface and use icons or images for faster recognition. Additionally, features such as the display of the course layout and the statistics that teachers can see with defined metrics such as percentage of correct exercises, engagement, and performance are presented.

A generalized data model was introduced with the two parts of student and content parts. It supported features, such as sequencing modules and lessons by time or exercise completion constraints.

The API requirements for good communication were outlined, including CRUD operations and query parameters such as pagination, filtering, and search. Further, the user permission roles consist of students and teachers, which can be subdivided into author and viewer. Students can get their courses and submit exercise progress. Teachers can create all described classes of the data model. However, they can only update their own and not get other teachers' occurrences, classes, students, and statuses.

Having obtained an understanding of the system's architecture, concepts, and requirements, the focus shifts to its implementation. Detailing decisions, specific implementations, challenges encountered, and resolutions in the subsequent Chapter 5.

Chapter 5

Implementation

Transitioning from the top-level overview in the previous Chapter 4, this chapter explains the system and its implementation. In Section 5.1, the adoption of Vue as the front-end (FE) framework is highlighted. Essential libraries, the course component, a responsive design, and the integration with external repositories are detailed. The chapter then progresses to the back-end (BE) in Section 5.2. The choice of Strapi as the CMS framework is explained, along with insights into the implementation of the data model, API configurations, and unit testing. Section 5.3 then describes an experiment undertaken to generate programming exercises using Chat GPT and the integration of Chat GPT into the system to assist teachers with creating programming exercises. This chapter's main points and findings are summarised in Section 5.4.

5.1 Front-end

This section presents the implementation of the FE. It starts with the reasons and motivations for using Vue as the FE framework for the teacher's UI, explored in Subsection 5.1.1. Subsection 5.1.2 presents the supplementary frameworks and extensions required during development. This is followed by Subsection 5.1.3, which explains how the course component was implemented, supporting the three roles of student, author, and viewer. The Subsection 5.1.4 presents the challenges faced in ensuring a responsive UX across diverse devices. Concluding the section, Subsection 5.1.5 provides insights into integrating external content. The import of JSON and CSV files with data such as student enrollments and the integration of AuthorKit, a repository of programming exercises, is explained.

5.1.1 Framework Selection

The web development ecosystem has experienced substantial growth in the number of tools and frameworks available for developers. Among the most prominent JavaScript frameworks for FE development are React, Angular, and Vue. Each offers a distinct approach to building web applications and has strengths and challenges.

Vue offers a set of methods, properties, components, and a design structure that's easy to grasp, making it an excellent choice for those new to framework-based development. Its reactive data binding and component-based architecture provide developers with the tools to create dynamic web applications. However, Vue is competing with more established competitors in the field. React, supported by Facebook, and Angular, backed by Google, have larger communities, resulting in more available resources, third-party libraries, and community-driven solutions. That said, Vue, with its growing community, is steadily trying to close this gap.

The choice of Vue for the Agni project came down to several factors. First and foremost, consistency was significant. With the student UI of Agni already using Vue, it made logical sense to ensure that both the student and teacher interfaces operated under the same framework. This not only ensures a unified experience but also streamlines the development process. Reusing components, logic, or styles from the student UI when building the teacher UI reduces redundancy and speeds up development.

Two main development strategies were under consideration: creating separate applications for the student and teacher UI or integrating both into a single application. The decision leaned towards developing one application for the entire FE. The reasons for this were twofold. Firstly, it simplified using and modifying the Agni student UI for teacher use. Secondly, the system wasn't large enough to necessitate two separate applications. Moreover, certain structures and data storage functions of the student UI could be reused for the teachers' UI.

5.1.2 Essential Libraries

When considering a FE framework, it's important to not just focus on the main development framework but also on the auxiliary tools that enhance and facilitate its use. As the core FE framework, Vue needs complementary libraries to realize its full potential. Vuetify, a popular Material Design component library, offers pre-built components that simplify development and ensure a consistent look and feel. However, Vuetify is also limited and sometimes can't satisfy an application's diverse requirements. Some functionalities, such as video display components, needed additional specialized libraries. This mixed approach, which combined Vuetify's component library with niche, specialized libraries, granted the flexibility to pick the best tool for each task.

Moreover, state management is a cornerstone of any dynamic application. The seamless interaction, data flow, and reactivity users expect from modern web applications can be attributed to adept state management. Vue Store (Vuex) is the official state management pattern and library for Vue applications. Vuex maintains a centralized store at its core, acting as a "single source of truth" for the application. This ensures that any data, once changed, reflects uniformly across components, eliminating discrepancies and potential data conflicts. For this project, Vuex played an important role in managing all kinds of data, such as user credentials, course data, and style configurations, which helped with the responsive design.

5.1.3 Course Component

An important concept highlighted in Section 4.2 was reusing the Agni student UI for teachers as an editable version. This approach allows teachers to preview edits in real time and offers an intuitive course creation and content modification interface. Therefore, the course component was refined to offer authors, viewers, and students different views. The general structure was kept, only modifying small elements, such as "spans" or "divs" containing texts, additional buttons, or icons. With the user login, the role, either as a student or teacher, is registered in the store. For teachers, this role is further updated into an author or a viewer when content is accessed and displayed in the UI. The different views of the course were implemented using Vue's "v-if" directive that conditionally renders an element based on the truthiness of its expression. If the expression for "v-if" evaluates to true, the element is rendered; if the expression evaluates to false, the element is not rendered. Figure 5.1 shows a code snipped of how a module's name is displayed in the header. Using the v-if directive, it is checked if the user is a student, viewer, or author, and depending on it, a div element with the computed name of the module or an editable component to write the module name is displayed. This approach preserves the core structure while displaying user-specific alterations.

```
<v-list-item-title :class="getTitleClass">
    <div v-if="isStudent || isViewer">
        {{ getModuleByResourceId(resource.id, resource.contentType).internalId }}.
        {{ getModuleByResourceId(resource.id, resource.contentType).name }}
    </div>
    <Editable
        v-if="isAuthor"
        type="module"
        field="name"
        placeholder="Module Name"
        :value="getModuleByResourceId(resource.id, resource.contentType).name"
        :id="getModuleByResourceId(resource.id, resource.contentType).id"
        @input="editableInput"
    />
</v-list-item-title>
```

FIGURE 5.1: Code for rendering the Module Name

5.1.4 Responsiveness

The Agni student UI exhibited several responsive issues at the project's beginning despite using Vuetify. For instance, buttons overlapped, and text sizes didn't adjust. Considering the variability of today's digital user base, ensuring that the interface is responsive to different screen sizes, from laptops and tablets to smartphones, was important. The goal was not just aesthetics; an interface that doesn't adjust properly can frustrate users, leading to a worse UX.

Vuetify inherently offers functionalities to make components responsive. The grid system employs a 12-point grid layout, with structured containers housing rows and adaptable columns. These columns are arranged to vary in size from mobile screens to expansive desktop monitors. With features such as breakpoints, offset, order, and auto-sizing, Vuetify offers developers a toolkit for responsive design. Further, breakpoints are customizable, allowing developers to tweak and adapt styles as required. These features were used in most of the teachers UI. However, the course component posed a unique challenge. While spanning the entirety of the student UI, this component only displays a segment of the screen in the teacher UI. The interfaces of the student and the teacher while creating a course can be seen in Figure 5.2. Therefore, directly making the component responsive to the overall screen size wasn't possible. To overcome this, the component's width is dynamically computed when the course component or sub-components are rendered or undergo a screen size change. This width is then captured in a variable stored within the Vuex Store module. Within this module, functions were defined to return style configurations based on the computed width. These functions are subsequently called to determine the style of the presented components. This approach ensures that the course component fluidly adapts its dimensions concerning the screen width, guaranteeing an optimal viewing experience across different devices.

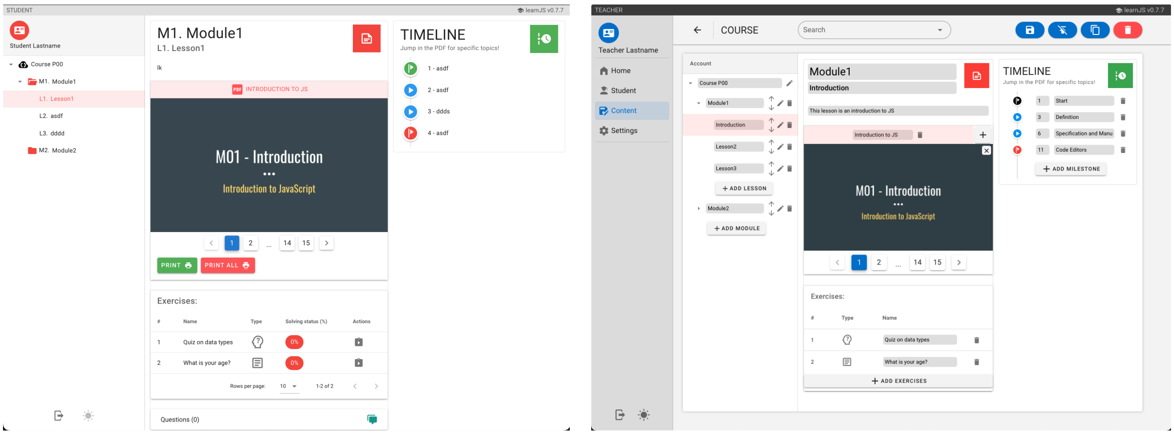

FIGURE 5.2: Agni Student UI (left) and Teacher UI to create a course (right)

5.1.5 External Content

This subsection explores the functionalities implemented to integrate external content. It includes importing students or classes via CSV or JSON files, eliminating manual entry, and facilitating the setup of classes for teachers. Moreover, a key objective of this thesis was to integrate external repositories to increase the number of available exercises and connect the system with other tools in the ecosystem.

Importing data, such as students and classes, simplifies a teacher's workflow, eliminating the need to create each element from scratch. The system supports importing student and class data with JSON and CSV files. While the system recognizes standard keys such as "studentName" or "className", the variability in key naming across different CSV and JSON files poses a challenge. Acknowledging this, after selecting a file, the system displays its native keys and those detected in the uploaded file in a dropdown selector for the teacher to match or verify them. Therefore, importing different formatted files is possible, avoiding manually creating each student and saving teachers time.

Shifting to the integration of external repositories to further broaden the resources available for teachers. AuthorKit, a tool for creating gamified programming exercises, was integrated into the system to enhance the range of available programming exercises. The choice of AuthorKit was driven by its big repository supporting many different kinds of exercises and easy, free accessibility. Its API provides the bridge for the integration. In AuthorKit, various information and configurations can be saved with an exercise. The integration prioritizes overlapping data, such as the exercise's name, statement, solution, and tests. Also, the import of only JavaScript exercises is possible. Further, AuthorKit normally contains dynamic input-output tests, which, depending on the solution of the exercise, were transformed into tests of type "log" if the solution returns an output with "console.log" or of type "func", if a function is contained in the solution. Given the flexibility offered by AuthorKit in exercise creation, the system can't cover every unique case. Therefore, after importing, the tests are run on the solution to verify its correctness and show the teacher possible errors. Moreover, exercises aren't immediately stored in the BE. Instead, they're held in the FE, for instance, within a course lesson. Only when the course is saved are the exercises stored in the BE.

5.2 Back-end

This section explores the BE and its implementation. It involves server-side responsibilities such as data management, processing, task execution, and system logic. Strapi, a headless CMS, was selected for this role. The section starts with discussing the reasons for choosing Strapi in Subsection 5.2.1. This is followed by exploring the implementation of the Data Model, aligned with Strapi's data structure in Subsection 5.2.2. Subsequent Subsection 5.2.3 explains customizations done to the API for guaranteeing wanted features and a better performance. The section ends by presenting the unit tests that reinforce key functionalities in Subsection 5.2.4.

5.2.1 Content Management System Selection

The creation of a BE system for a project presents numerous options. It could involve a full development process using BE frameworks such as Node.js with Express.js, NestJS, or Python with Django or Flask. Alternatively, Backend as a Service (BaaS) services such as Firebase, Parse, or Back4App could be employed. Another option is a headless CMS such as Strapi, Contentful, or Prismic.

A headless CMS has several advantages compared to other options. It provides a quick and minimal coding setup of a BE system with key features such as user authentication oftentimes included. It also has a built-in user interface for creating and managing content. It also provides an API for integrating an UI to display or even manage and create content. Nevertheless, using a headless CMS has drawbacks. These include limited control over the BE logic and UI for content creation. Further, potential cost implications are associated with some CMS platforms. Despite these drawbacks, the benefits outweighed the cons, leading to selecting a headless CMS as the BE service.

Several headless CMS options are available, including Contentful, Sanity, Ghost, Prismic, and Strapi. In comparison to its counterparts, Strapi presents a few challenges. It has a relatively younger community and may lack some features provided by more mature CMS platforms. Furthermore, being built on Node.js could impact its performance in large-scale, high-traffic applications compared to CMSs developed in more efficient languages. However, Strapi also offers advantages. As an open-source platform, it is freely available. It is easy to set up and developer-friendly, making it an attractive choice. Moreover, it offers the customization of the API. Considering these advantages and the challenges, using Strapi for the BE service was also predetermined.

5.2.2 Data Model

Strapi's data structure can be categorized into three distinct types: collection type, single type, and component, as previously detailed in Section 3.1. To briefly refresh, Single and Collection types are independent, each with its own API endpoint. While Single types are limited to one instance, Collection types can manifest in multiple instances. Components, on the other hand, are reusable entities that exist within either a collection or a single type.

A generalized data model and its intentions were detailed in Section 4.3. Figure 5.3 translates the data model into the Strapi data structures, highlighting its principal classes. The complete BE data model is shown in Appendix B. The distinctions between green Components and blue Collection types are displayed. For the content aspect, `Course`, `Expositive`, `Evaluative`, and `Question` have been implemented as Collection Types. This allows educators to access, create, and edit them individually, making them easy to reuse. Similarly, `Occurrence`, `Class`, `Student`, and `Status` have been selected as Collection Types for easier creation and accessibility. The other entities, such as `Modules Lessons` etc., which are nested within courses, are categorized as Components since there's no need to create or modify them individually.

This data model was implemented using the built-in UI of Strapi to create a data structure, as shown in Figure 3.3.

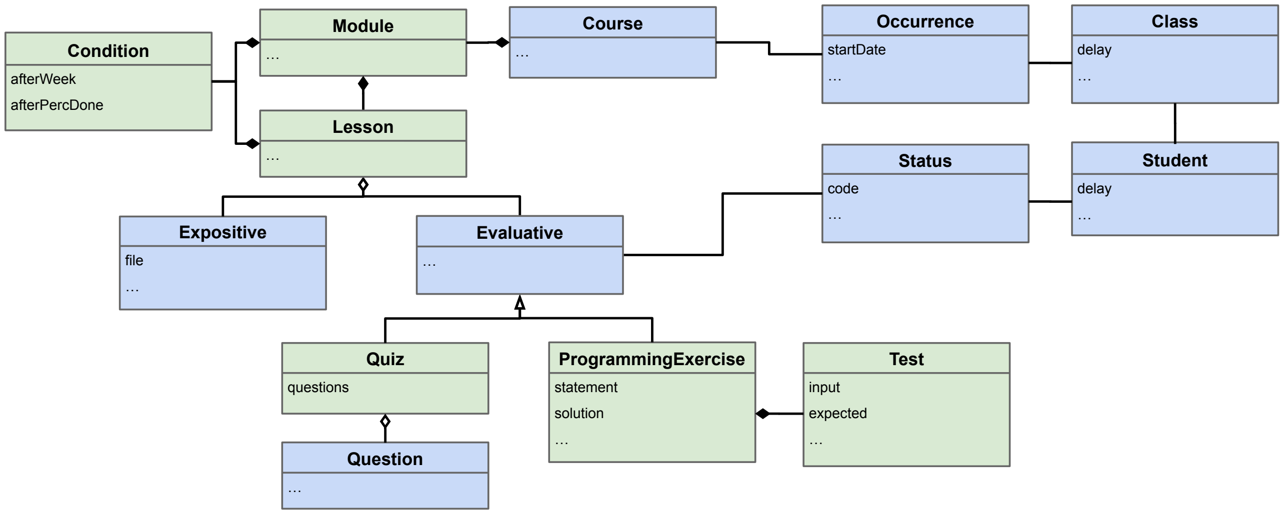

FIGURE 5.3: Strapi Generic Data Model

5.2.3 Customization

Some modifications had to be made to Strapi to facilitate interaction between API and improve data management performance. Including the implementation of hooks to customize the response of certain requests, a new endpoint to search in all collection types, and the change of the database management system to PostgreSQL.

Strapi's default configuration only allows the creation of a single collection type with one request. Given this limitation, creating a complete course with other collection types such as *Expositives*, *Evaluative*, and *Questions* would have required multiple individual requests. To address this issue, collection hooks were employed. In essence, implementing a hook customizes the standard behavior of requests. Therefore, creating an entire course, evaluatives, occurrences, and classes was implemented. For instance, when creating a course and introducing a new quiz, all relevant data is submitted to the course endpoint via a singular post request. Following this, a hook for the course calls the evaluative hook with the respective data, which, for a quiz with new questions, subsequently triggers the question hook to create that question. This chain of internal calls significantly reduces the number of requests between the FE and BE. Beyond this, creating multiple instances of each collection type with a single request, making content creation more efficient, was implemented. Other functionalities, such as cloning a course with the flexibility for teachers to specify modules, lessons, and evaluatives or expositives they wish to replicate, have been implemented using collection hooks.

The roles of Student and Teacher, which can be further defined as Author and Viewer, were implemented in two different ways. The distinction between Student and Teacher is made with the user-permissions plugin supported by Strapi. When creating a user, the specific role is saved with him. These roles and the respective permissions of which requests one can do were configured using the UI of Strapi. Whenever a collection type is created, the author is saved. This is implemented through the collection hooks. When a teacher requests data or wants to update content, it is verified whether the request comes from the author or a viewer (non-author). Different responses are triggered depending on it, such as differences in the data returned or denying the request.

A custom endpoint named ”content” was introduced to provide a complete search functionality across all entities. Therefore, teachers can search across various collection types by text, including courses, exercises, students, and more. It delves into different fields, searching through goals, module names, exercise titles, etc. The results from these collection types are then aggregated and returned in a single response. Moreover, the endpoint allows for a specialized search highlighting the latest published content, spanning courses, expositions, evaluations, and questions. This provides teachers with updates and new content they can check out in the home menu.

Strapi's default configuration for database management is SQLite. Its simplicity makes it suitable for quick prototyping. However, SQLite's limitations became apparent as the project grew in complexity. In particular, its struggles with scalability and inability to efficiently handle write operations posed a problem. When considering alternatives, three relational database management systems (RDBMS) supported by Strapi emerged as contenders: MySQL, MariaDB, and PostgreSQL. MySQL, one of the most popular open-source databases, is known for its speed and reliability. Its wide user base provides good community support. However, when it comes to certain features, such as support for JSON fields or handling transactions efficiently, MySQL has its limitations. MariaDB, an evolution of MySQL, is designed to maintain compatibility with its predecessor while introducing new features and enhancements. While having improvements over MySQL, MariaDB carries many of MySQL's inherent structural and performance characteristics. PostgreSQL stood out due to its extensibility and robust handling of relational databases. Its native support for JSON, advanced data types, and reputation for managing relational data models effectively made it the preferred choice. The migration from SQLite to PostgreSQL in Strapi is easily handled by changing and adding some configurations. In practice with PostgreSQL, operations such as updating or creating a small course, which took about 1-2 minutes with SQLite, were reduced to a few seconds. This change was crucial in ensuring sufficient speed in handling important requests.

5.2.4 Unit Tests

For the fundamental functionalities, as well as most of the implemented hooks, unit tests were incorporated. Unit tests play an important role in software development, ensuring that individual units of the software work as designed. Their primary purpose is to validate the smallest parts of the application in isolation, guaranteeing that they function correctly under various scenarios. Jest, a standard tool for integrating unit tests in Strapi, was used for this purpose. When writing these unit tests, attention was set to testing the customization and collection hooks implemented and verifying key functionalities of the system. Both positive tests, which checked for expected outcomes, and negative tests, designed to induce and handle errors, were developed to offer a full spectrum of validation scenarios. During the testing phase, a separate empty database is created. This allows real data to be created, read, updated, and deleted, offering a testing scenario while confirming that stored data remains unaffected. Unit tests were specifically developed to evaluate the creation, updating, reading, and deletion processes for all elements in the system. These include courses, expositives, evaluatives, questions, occurrences, classes, students, and statuses. While it's always possible to encounter issues within the API, these unit tests provide a robust assurance of system functionality under normal conditions. They validate the reliability of the implemented features and ensure the system operates as expected, enhancing its overall integrity.

5.3 Assisted Exercise Generation

Creating exercises is an important but time-consuming and error-prone task when it comes to developing learning content for programming. Therefore, one way to enhance the UX for teachers is to create tools that assist them in this task. AI language-based generative models, such as chat GPT, have created an opportunity to do this.

This section focuses on assisted exercise generation using GPT. It starts with Subsection 5.3.1, which describes an experiment of automatically generation programming exercises using GPT. Subsection 5.3.2 then explains its integration into the proposed system.

5.3.1 Experiment

An experiment was conducted to create programming exercises using the GPT-3.5 API by OpenAI. This experiment was done within the project Framework for Gamified Programming Education (FGPE)+ with the objective of generating 120 programming exercises for AuthorKit. AuthorKit allows for the definition of many different parameters for an exercise. For the experiment, the desired parameters for each exercise were title, difficulty, context, task, input description, output description, example, solution language, solution code, and five input/output tests. To experiment more with GPT's capabilities and as the project FGPE+ involves universities of different countries, the generation of these parameters in different languages, including Portuguese, English, Italian, and Polish, was decided. The generated exercises should then be converted into the appropriate format of AuthorKit, YAPExIL, and added to the repository. The GPT-3.5 model was tasked with inventing the exercises independently, without relying on external descriptions.

The experiment posed significant challenges, particularly in generating consistent and reliable responses that contained the specified parameters and could be automatically converted. The initial attempt involved a simple prompt, instructing the model to generate programming exercises considering the desired parameters. In the Appendix C Section C.1 can be seen the first prompt. However, this simple prompt got a huge inconsistency in responses, such as variations in the use of code fields, headers, subheaders, list, markdown, or table formations, and alterations in parameter names, which made automatic conversion impossible. Several modifications were explored to mitigate these issues, including providing explicit instructions to the model on unwanted formats and presenting examples of the desired "key: value \n" structure. Despite these modifications, the differences in format and key names continued. The most effective resolution found was to instruct the GPT-3.5 model to answer in a JSON format, accompanied by a format example. This strategy delivered the most consistent answers and uniform key names. The final prompt to generate the exercises can be seen in the Appendix C Section C.1.

Using this prompt, the desired 120 programming exercises were generated. Each response was then automatically converted into an object to test the validity of the JSON format and the input/output tests with the solution code. Multiple errors were encountered throughout the process, such as JSON errors indicating an invalid format and code errors appearing when an input/output test failed. When an error emerged, another prompt was sent to describe the type of error and ask GPT to resolve it. Example responses of a correct exercise and exercises with JSON and code errors are displayed in the Appendix C Section C.2.

	Total	% of Total	Resolvable	% of resolved
Correct Exercises	120	21%		
JSON Errors	199	35%	23	12%
Code Errors	245	44%	3	1%
Total	562	100%		

TABLE 5.1: Results of Automated Exercise Generation with Chat GPT

Table 5.1 outlines the results of generating the 120 exercises. To obtain these, it was necessary to generate 564 exercises. Therefore, only 21% were correct. 199 exercises (35%) contained JSON formatting errors, such as missing commas or brackets. 12% of these, amounting to 23 exercises, could be corrected by pointing out the issues to GPT. Furthermore, 245 exercises (43%) presented code errors, implying errors either in the solution code or the input/output tests. However, only 1% of these, specifically three exercises, could be corrected with the assistance of GPT. Correct exercises were manually verified if the generated parameters fit together. Thereby, no discrepancies were found.

These results point to several issues, starting with the high occurrence of JSON errors. As previously discussed, establishing a consistent format posed a considerable challenge. Despite asking for the answer in a JSON format, GPT struggled to create it correctly. The same issue was found by other programmers mentioned in the community forum of OpenAI [65]. Therefore, it points to a general limitation of GPT. Furthermore, the fact that 43% of the generated exercises contained errors in input and output tests or in the solution code underscores the difficulties GPT faces in generating fully correct code. A possible solution to mitigate these issues could be to generate the parameters step-by-step through a conversational approach with GPT. It is said by multiple websites that a step-by-step approach should lead to better results [66, 67]. However, when using the API, the number of requests per minute is limited to 3, which was why this method was not pursued. Other improvements could be by providing more context and a specific role to GPT, which is said to have a positive impact.

A possible limitation of this experiment could have been the use of GPT version 3.5. Version 4.0 was already available but was not chosen in this context because of the higher cost. This experiment handled the creation of exercises invented by GPT. The feature integrated into the proposed system gives teachers the ability to describe the exercise they want to generate, therefore not being the exact same situation. However, due to the consistency between the generated parameters of an exercise, this should not significantly change the results obtained from this experiment.

This study shows GPT's limitations with generating code and a consistent JSON format. 35% and 44% of JSON and code errors are significant. Despite the mixed results, the integration of GPT was seen as an innovative addition to the system. However, the approach needs to be improved to get more resistant results.

5.3.2 Implementation

The implemented feature allows teachers to write brief descriptions of programming exercises they want to be generated. Therefore, a text area is displayed in the interface for creating a programming exercise. The description will be sent to the BE, where another request, including the specific prompt, is sent to the OpenAI API to generate the exercise. For the generation, GPT-3.5 was used due to its reduced cost. Also, adjustments were made to the methodology presented in Subsection 5.3.1. The approach recommended by Joseph Martinez [68] was followed. Instead of specifying a JSON format in the prompt and providing an example, a JSON schema is passed as parameters to define the desired response format. Additionally, the role of a teacher wanting to assist the user has been assigned to GPT. The parameters wanted in the generation are an exercise name, statement, solution, and 3 test cases. The code for sending the prompt is displayed in the Appendix C Section C.3.

Using this approach, the responses are more consistent with valid JSON formats. Nevertheless, it is being verified, and if successful, the parameters are written into the UI. The exercise, with its tests, is automatically run in the FE to identify and show potential problems with the solution code or test cases. With the high percentage of code errors found using GPT in the previous subsection, the feature is only an assistant for the teacher and cannot be handled as completely reliable.

5.4 Summary

This chapter delved into the implementation of the system and its features. The chapter starts with the FE, describing the decision to use Vue. This choice primarily aligned the teacher and student UI for Agni with the same framework. Key libraries such as Vuetify2 and VueStore streamlined the implementation. Subsequently, the approach to use the student UI and transform it into an editable interface is detailed. While the main component structure remained, Vue's "v-if" directive defined different views. This course component, visible to teachers and students with different occupations on the screen, created a new challenge of making it responsive. The solution was a dynamic computation of the component's size, on which style configurations were based. Lastly, importing data from files, such as CSVs or JSON, or integrating AuthorKit to enrich the exercise pool was implemented.

Next, the focus shifted to the BE, discussing the selection of Strapi. Its main reason is flexibility, cost-effectiveness, and ease of use as a headless CMS. The implementation of the data model in Strapi is also elaborated on, shedding light on the use of content structures such as collections and components. The decision on which data structure to use was influenced by whether a class required independent creation, updating, deletion, and an API endpoint. The use of collection hooks to reduce the number of requests when creating a course is presented, and the significant change from SQLite to PostgreSQL significantly improved the system's speed. Finally, the section concludes by discussing the implementation of unit tests on the Strapi API. These tests ensure the functionality and enhance the system's integrity.

The last section of this chapter presents the assisted exercise generation using the GPT model. An experiment of creating a programming exercise with GPT-3.5 showed its limitations in creating valid JSON formats and fully correct programming exercises where the solution and test cases provided are correct. To solve the JSON errors, the integrated system followed a different approach of detailing the wanted response format in the request parameters in a JSON schema. Further, when receiving a valid exercise, its solution is run on the test cases in the FE to notify the teacher if errors occur.

Having detailed in the preceding two chapters, the system's architecture, concepts, and implementation, as well as the pivotal decisions and challenges addressed in the process of developing the proposed system. The subsequent Chapter 6 will explain the evaluations conducted on both the Strapi UI and the developed UI, examining and discussing the outcomes.

Chapter 6

Validation

This chapter explores the validation of the main goal of this thesis to propose a system with a good UX. During the project, two evaluations were undertaken. The first assessment targeted Strapi's UI, measuring the UX when using it to complete teacher tasks. The unsatisfactory results led to developing a new UI. This UI, including the key functionalities such as content and student management, underwent another evaluation. The chapter begins by explaining the chosen evaluation methodology, as outlined in Section 6.1. It then presents the evaluation of the Strapi's UI, encompassing the findings, a discussion, and conclusions in Section 6.2. The following Section 6.3 explores the evaluation of the developed UI. The results are presented, discussed, and compared to the results of the first evaluation. The chapter ends with a recap, presenting the discoveries and takeaways from the evaluations in Section 6.4.

6.1 Evaluation Methodology

Before presenting the two evaluations of this thesis, it's essential to understand the rationale for selecting a questionnaire grounded on Jakob Nielsen's usability heuristics.

Several methods are available for evaluating the UI, ranging from expert evaluations and task-based performance measures to questionnaires. In this context, questionnaires were chosen as the primary evaluation tool to get insights into external users' opinions. While the developers have their perspective on the system, often, there exists a divergence between these views. Furthermore, questionnaires ensure respondent anonymity, thereby encouraging honest feedback. Compared to in-person usability testing or expert evaluations, they are a more economical and scalable choice, free from the needed equipment or venues. Questionnaires allow participants to provide feedback at their own pace, which can, on the one hand, lead to good responses but also to a poorer quality of responses if it is not taken seriously.

There are different questionnaires, as detailed in Subsection 2.1.3. Among the most popular ones are AttrakDiff, UEQ, and SASSI for UX evaluation. There are also usability questionnaires anchored on Jakob Nielsen's 10 usability heuristics for user interface design. Given this project's emphasis on crafting an intuitive UI and wanting more insights into the possible usability problems and improvements, further not having many evaluators available, Nielsen's approach and a questionnaire based on his 10 usability heuristics were chosen. To these heuristics were added easiness, speed, and reliability of key functions. Each category contains two to four questions with the answer possibilities ranging from "Never" to "Always" with the apart option of "not applicable". For example, questions for the heuristic Aesthetic were:

A Is the information on the screen just what I need?

B Does the information on the screen stand out from the background?

C Aesthetically, is the system pleasant in terms of: colors, brightness, etc.?

The questionnaire further contains a system classification, graded from "Bad" to "Very Good". After that, respondents can answer open-ended questions about the system's strengths, weaknesses, and improvement suggestions. Before answering the questionnaire, respondents executed standard teacher tasks, such as course creation, material reuse, and class creation, ensuring they had hands-on experience with the system's functionalities. These tasks are displayed in the Appendix D Section D.1. The questionnaires for the two evaluations can be seen in Section D.3 and E.2 of Appendix D and E, with slight differences in the questions regarding Reliability.

6.2 Strapi User Interface Evaluation

This section explores the evaluation of the Strapi UI, including its results, discussion, and conclusions. The purpose of evaluating the Strapi UI was to determine its sufficiency in delivering a satisfactory UX. For this assessment, the evaluators were five master's students, aged between 20 and 25, from the University of Porto. These individuals are enrolled in courses related to informatics, such as computer science, data science, and statistical analysis, and include two women and three men. They were assigned typical teacher tasks and later responded to a questionnaire, shown in Appendix B Section D.3.

The UI of Strapi version 4.4.5 was evaluated, including the implemented data structure of Section 4.3. Example images of this UI can be seen in Appendix B Section D.2.

Category	%	Mean
1. Visibility	80	3.25
2. Compatibility	90	3.17
3. Freedom	70	3.14
4. Consistency	80	3.94
5. Prevention	45	3.11
6. Emphasis	70	3.93
7. Flexibility	67	2.00
8. Aesthetics	80	3.25
9. Help to Users	67	3.20
10. Help with documentation	53	3.25
11. Easiness	80	2.25
12. Speed	50	4.20
13. Reliability	100	2.91
14. Classification	100	2.80

TABLE 6.1: Strapi UI Evaluation Results Table

Table 6.1 displays the evaluated heuristics, with the percentage of questions answered by the respondents and the mean score of their responses. In the Appendix Section D.4, the responses by every respondent to all questions, with their percentage of answers and score, are displayed. Figure 6.1 visualizes the score of every metric with a bar chart ordered decreasingly. As mentioned in Section 6.1, the respondents answered with "Never", "Almost Never", "Regular", "Almost Always", and "Always", which were transformed into 1 to 5 to calculate the scores.

Analyzing the results, it can be noted that most of the heuristics were evaluated positively. Especially `Speed`, `Consistency`, and `Emphasis`, with a score of about 4, were the highest. However, `Speed` with a 4.20 score had only 50% of questions answered, meaning many of them were thought to be not applicable. On the other hand, `Flexibility`, `Easiness`, `Reliability`, and `Classification`, with a score of 2, 2.2, 2.9, and 2.8, respectively, were evaluated negatively. `Flexibility` also had a relatively low percentage of questions answered, 67%. `Help with documentation` and `Prevention` had a low percentage of questions answered, specifically 53% and 45%. As strong points of the system were mentioned, the easy creation of a course with modules and lessons, excluding expositives and evaluatives, which were mentioned as a weak point. The need to create collection types such as courses, expositives, evaluatives, and questions individually and only after that being able to associate them with each other was seen as unintuitive and time-consuming. Another strong point was the generally easy-to-use and fast UI. Further weak points were a lack of a help system, a landing page without helpful information, and the lack of explanation of some fields. A tutorial was often mentioned as an improvement suggestion.

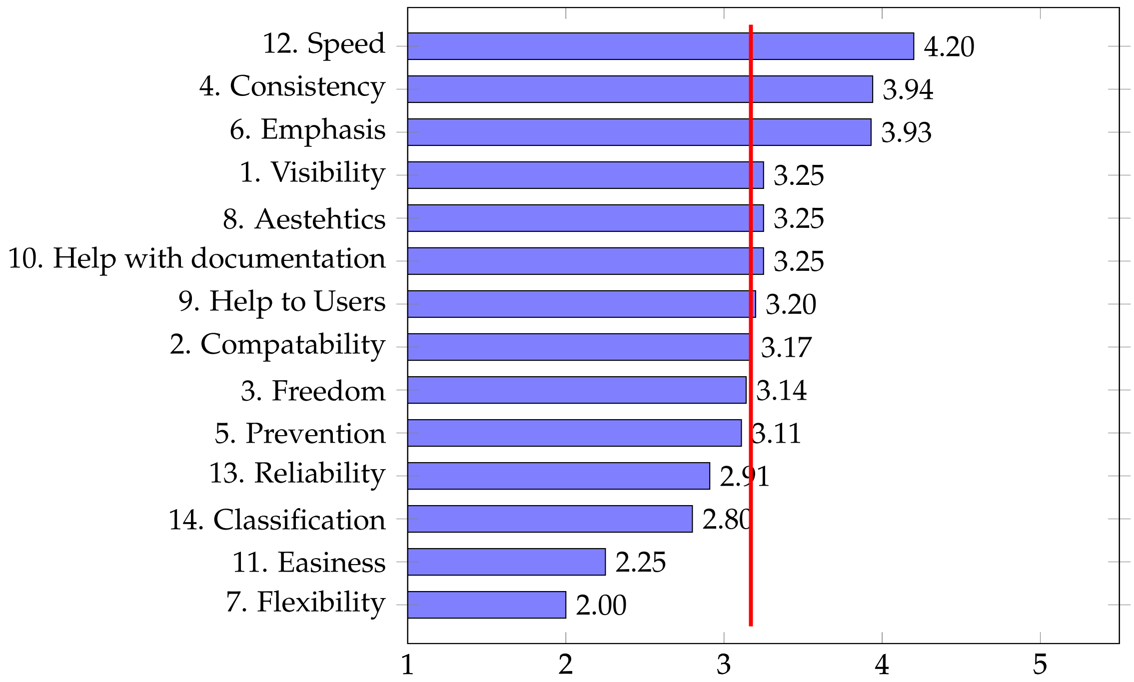

FIGURE 6.1: Strapi UI Evaluation Results Chart (ordered by score)

The overall more positive evaluation of the 10 usability heuristics for Strapi's UI was expected due to the UI being one of the positive aspects of why people use Strapi. Strapi is very fast in all operations, and with intern evaluations and a design team behind the UI of Strapi, the positive evaluation of `Speed`, `Consistency`, and `Emphasis` seems congruent. `Aesthetics`, as is of a subjective quality, also seems appropriate, with a 3.25 score. With no real help and explanations in the main menu, some confusion of the evaluators on how to use the system, as well as no tutorial or guide on how to use the system, the more neutral evaluation of `Visibility`, `Help with documentation`, `Help to users`, and `Compatibility` can be reasoned. However, the evaluation of `Freedom` and `Prevention` seems lower than expected. Strapi validates all operations and prevents users from making errors. Also, it gives users the freedom to interrupt and continue actions. The difficulties found in creating a whole course with each collection type separately explain the more negative scores in `Reliability` and `Easiness`. `Flexibility`, with the lowest score of 2.00, also seems understandable due to Strapi not having options for customization.

This validation has limitations derived from the evaluator's profile and their number. While student participants provided valuable feedback on the UX, teachers were the pretended choice for evaluators. Due to certain issues, they couldn't be included. Teachers have a deeper understanding of the specific needs and demands associated with their roles within such systems. Moreover, the evaluation was conducted with only five evaluators. While Jacob Nielsen points out that this number is sufficient to identify most usability issues, there are counterarguments, as discussed in Subsection 2.1.3. Some studies question Nielsen's method, proposing that a group of 10 evaluators might be more effective. Engaging 10 participants instead of 5 could enhance the study's robustness without undermining reliability. It's also important to note that the evaluators' familiarity with the author might have introduced some bias.

Despite the mentioned limitations, the study highlighted that Strapi's UI, in managing course content and facilitating student interactions, is not intuitive. This is especially shown in the low `Reliability`, `Classification`, and `Easiness` scores. Given its non-customizable nature, refining its UI is impossible. As a result, the decision was made to develop a new UI for teachers while retaining Strapi's API and content storage for back-end operations.

6.3 Final User Interface Evaluation

This section presents the evaluation of the final UI, containing the key features but not all other described ones. The same five students from the initial assessment participated, joined by five professors from ESMAD P.Porto and the University of Porto, who have experience with e-learning systems as educators. These five professors include one woman and four men between the ages of 30 and 60. The students, familiar with the initial evaluation, were reminded to compare the new UI with the Strapi UI. They were allowed to revisit the Strapi UI for reference, facilitating a direct comparison with the newly developed UI. They were also provided with their initial responses to ensure continuity in their assessment. Meanwhile, the five professors centered their evaluation solely on the new UI, without the backdrop of the initial assessment. All the evaluators were given the same tasks as in the first evaluation with a slightly different questionnaire. Some questions unsuitable to the final UI were removed from the reliability. The complete questionnaire can be seen in the Appendix Section E.2.

As stated before, the evaluated UI didn't include all features described in this thesis. It supported only student and course management. It didn't include statistics, course layouts, features to import files, or integration with AuthorKit or GPT. The search through all collection types, the settings, and the account menu, with logout and dark mode, were also not yet included. Screenshots of the interfaces of the evaluated UI can be seen in Appendix C Section E.1.

Table 6.2 shows the evaluation results by category. It displays the overall results, highlighting the percentage of questions answered and the score. These metrics are broken down into the groups of students and teachers. As in the previous section, response options "Never," "Almost never," "Regular," "Almost always," and "Always" were numerically converted, ranging from 1 to 5. For a visualization of the data, Figure 6.2 shows the scores by students, teachers and all evaluatores. Looking at the overall scores, all categories, except `Help with documentation` and `Flexibility`, are around and above 4. `Help with documentation` and `Flexibility` are more neutral evaluated around 3. `Speed`, `Consistency`, and the `Classification` having the top 3 overall scores. The chart also shows the difference between the students' and teachers' scores, with the teachers always evaluating the system worse than the students. The biggest disparity is in the `Classification`, `Easiness`, and `Flexibility`. `Flexibility` was also only evaluated with 37% of the questions. Teachers' worst evaluated categories were `Easiness`, `Help with documentation`, and `Flexibility`. In contrast, students worst evaluated categories were `Flexibility`, `Help to users`, and `Help with documentation`. In the open-ended questions, evaluators frequently highlighted the system's simplicity and intuitiveness as its strengths. However, they also pointed out weak areas, including initial challenges faced during usage, occasional errors, and the absence of confirmations and customization options. For improvements, suggestions included introducing a tutorial, revisiting the testing and refinement process, improving the onboarding experience, enabling the import of CSV or JSON files, and incorporating auto-save and hotkey features.

Upon examining Figure 6.2, a noticeable disparity between the assessments of teachers and students immediately stands out. This could be attributed to students being instructed to compare the developed UI with the Strapi UI, which they previously critiqued

Category	All		Students		Teachers	
	%	Mean	%	Mean	%	Mean
1. Visibility	93	3.95	85	4.41	100	3.55
2. Compatibility	100	4.20	100	4.60	100	3.80
3. Freedom	60	3.96	55	4.18	65	3.77
4. Consistency	80	4.50	80	4.88	80	4.13
5. Prevention	65	3.85	60	4.42	70	3.36
6. Emphasis	85	4.06	75	4.53	85	3.68
7. Flexibility	37	2.82	27	4.25	47	2.00
8. Aesthetics	100	4.27	100	4.53	100	4.00
9. Help to Users	90	3.96	87	4.15	93	3.79
10. Help with documentation	73	3.14	73	3.55	73	2.73
11. Easiness	100	3.95	100	4.70	100	3.20
12. Speed	80	4.63	90	4.89	70	4.29
13. Reliability	99	4.30	100	4.71	98	3.89
14. Classification	100	4.40	100	5.00	100	3.80

TABLE 6.2: Final UI Evaluation Results Table

as lacking. Teachers also have a clearer understanding of their needs and are more experienced with such systems, having a more critical eye. The categories of `Flexibility` and `Help with documentation` received the lowest scores. This is understandable, considering the evaluated system did not offer tools for interface customization or user assistance at that stage. A big difference in perception between students and teachers emerged in the areas of `classification`, `Easiness`, and `Flexibility`. Teachers found the system more challenging to grasp than students, who provided notably favorable evaluations for both `Easiness` and `Classification`. Further, the disparity in `Easiness` could come from the students considering the time-consuming and unintuitive way of creating courses with Strapi. Their major critics of having to create exercises and expositives independently and having no organization of the collection types were resolved. Teachers also identified issues with `Visibility` and `Error prevention` compared to the other categories, likely reflecting the system's initial lack of data verification. `Speed` and `Consistency` were the highest-rated categories, with the shift from SQLite to PostgreSQL majorly contributing to the positive rating on `Speed`. The three-part design structure and the consistent appearance achieved using Vuetify might have boosted the `Consistency` scores. Interestingly, one teacher's evaluation stood out as an outlier. This teacher found the system counterintuitive and rated many criteria unfavorably. Such evaluations underscore the challenges of designing a UI that pleases everyone. While the general sentiment is positive, it's crucial to acknowledge critical feedback. This can help pinpoint improvement areas and accommodate the broadest range of users.

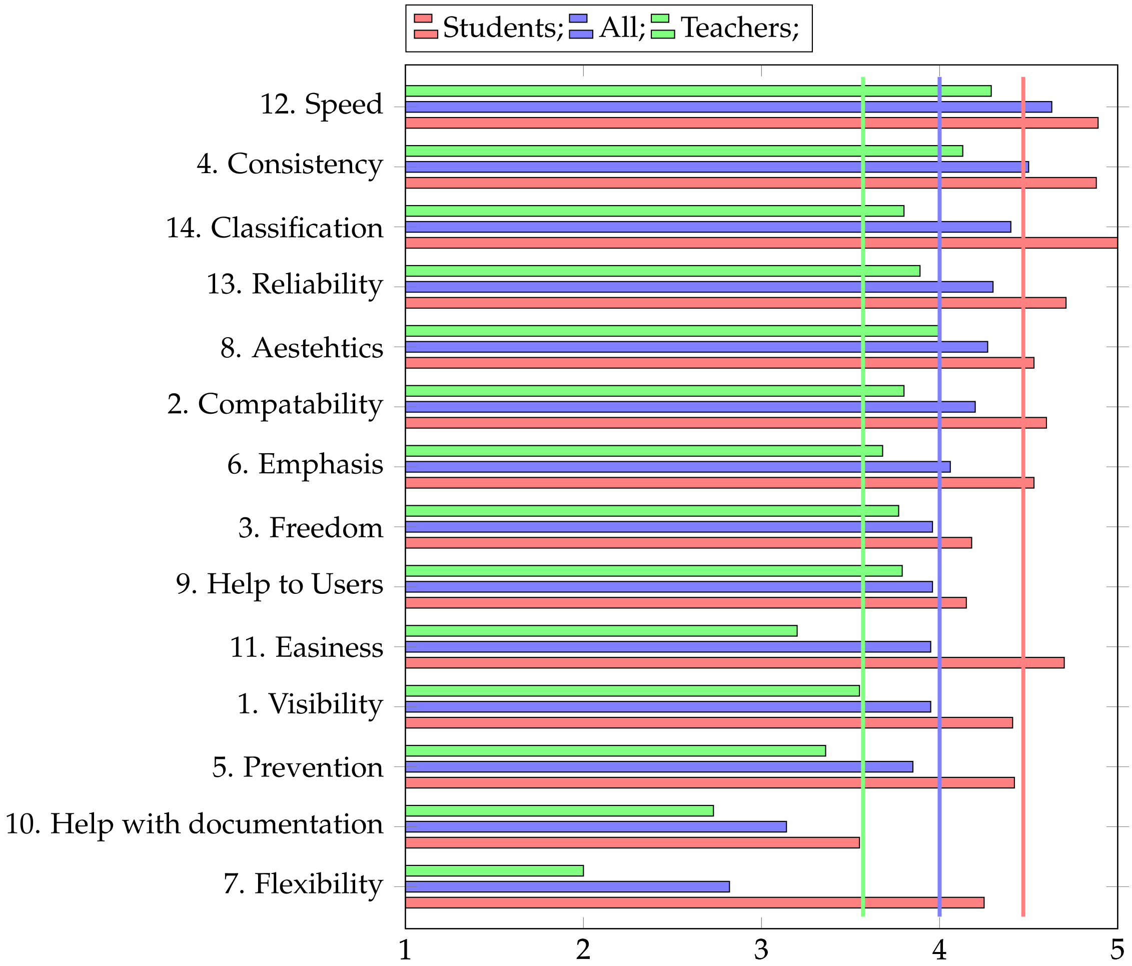

FIGURE 6.2: Final UI Evaluation Results Chart (ordered by score of all evaluators)

Previously, the Strapi evaluation was conducted with five students, who later assessed the developed UI. These students were reminded about their previous evaluation and provided with their earlier feedback on the Strapi UI. They were encouraged to keep the Strapi UI in mind while evaluating the new Agni UI. Figure 6.3 contrasts the evaluations of the Strapi UI and the developed UI by the same students. In the figure, the red bars represent the Strapi UI scores, while the blue bars denote the scores for the developed UI. Categories are arranged in descending order based on the score difference between the final UI and the Strapi UI. At first glance, the developed UI outperforms the Strapi UI across all categories, indicating significant UI advancements. Categories such as `Flexibility`, `Easiness`, `Reliability`, and `Classification` show improvements by roughly 2 points. The students' earlier criticism of the Strapi UI's unintuitive collection type creation process was addressed, introducing the reuse of the Angi student UI in an editable version.

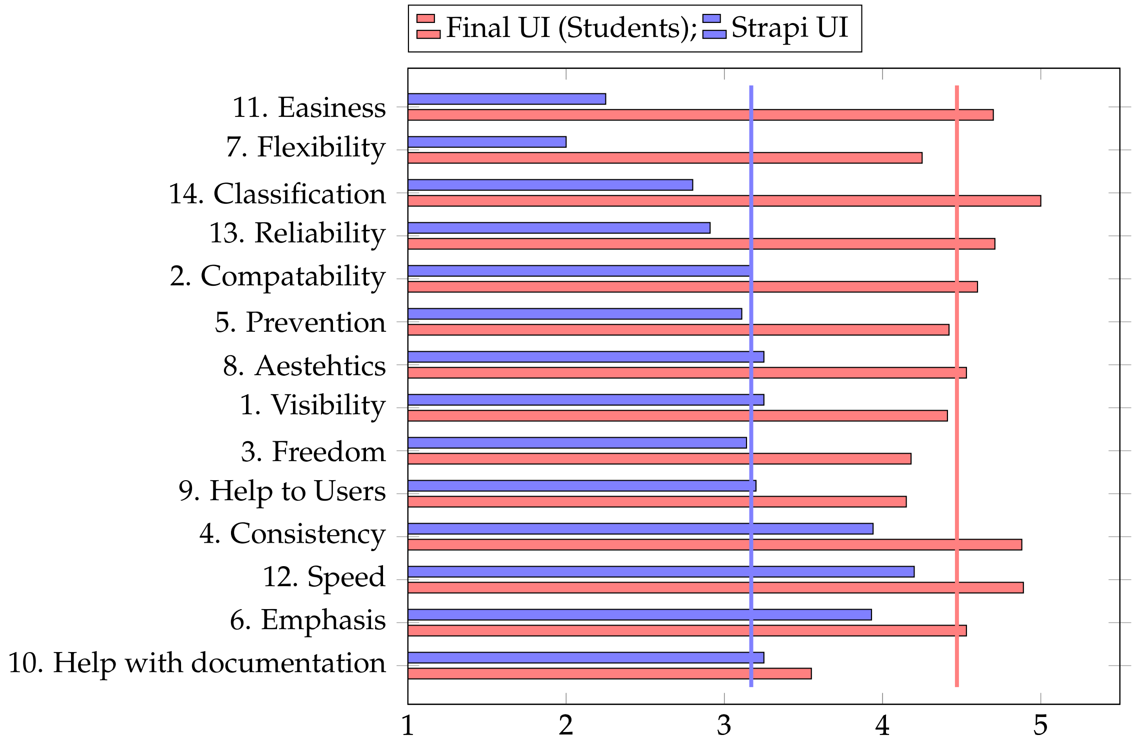

FIGURE 6.3: Comparison of the Strapi and Final UI Results (ordered by difference)

The newly introduced hierarchical structure with content and student management seems to have enhanced the system's navigability. These enhancements appear to be successful, especially since the strengths of the second evaluation include intuitiveness and simplicity, areas once described as weak points in the Strapi assessment. Other areas, such as `Compatibility`, `Prevention`, `Aesthetics`, `Visibility`, `Freedom`, `User Assistance`, and `Consistency`, also show marked improvement, albeit with lesser differences. While Strapi had superior error prevention mechanisms, the overall improvement in scores could be reasoned by the higher easiness of use, improving the satisfaction with the developed UI, which might account for its uniformly better scores across all categories. The smallest disparities in scores were found in `Speed`, `Emphasis`, and `Help with Documentation`. The similar scores in `Help with Documentation` between the two systems can be explained, given that neither system substantially aids users with guidance or support features.

The shortcomings of the initial evaluation, especially the lack of teacher participants, were addressed in the second study, which also expanded the size of the evaluator pool. However, the potential bias arising from the author's familiarity with the five student evaluators remained consistent in both evaluations.

Despite certain limitations, there's a significant progression from the Strapi UI to the developed UI. This is particularly evident in key areas such as easiness, classification, and the reliability of critical functions. Although more critical, including teacher feedback still casts the developed system positively. Implemented enhancements, such as the ability to import classes and students using JSON or CSV files, address specific suggestions for improvement. Errors identified have been fixed, and the interface has been subtly refined, particularly in the appbar's display. Other improvement suggestions and weak points were resolved, and missing features, including the integration of Authorkit and GPT, the addition of statistics, and a general search function, among others, were implemented.

6.4 Summary

This section presents an important aspect of UI development: the validation. The selected validation methodology was a questionnaire rooted in Jakob Nielsen's 10 usability heuristics, selected for its good insights into usability, potential issues, and points for enhancement.

Initially, the Strapi UI underwent evaluation, revealing unsatisfactory results. It was perceived as time-consuming and unintuitive. Notably, `Flexibility`, `Classification`, `Easiness`, and `Reliability` were highlighted as significant weak points, with three of them being identified as pivotal. The resolution to design a new UI emerged from these results and Strapi's customization limitations.

Subsequently, an evaluation of the developed, albeit not final, UI was undertaken. The evaluated UI contained the key functionalities. The same evaluators of the Strapi assessment found the developed system significantly better and outperformed it in every category. `Flexibility`, `Easiness`, `Classification`, and `Reliability` were areas that were evaluated negatively in the Strapi evaluation and showed the highest improvement in scores. Incorporating the perspectives of the teachers further strengthened the credibility of the findings. While these teachers approached the system more critically, their overall assessment was still favorable, suggesting that the developed system provides a positive UX.

Chapter 7

Conclusion

This thesis focused on enhancing the UX for teachers in a VLE. Specifically, it presents the evolution and completion of the Agni Web Playground, with the missing part of a teacher's UI and a back-end system to process and store data. The study began by surveying the current UX and e-learning systems research landscape in Section 2.1, 2.2. Subsequently, in Section 4.1, 4.2, 4.3, and 4.4 concepts and strategies related to the UI design, Data Modeling, and API construction were outlined to provide an insight into the system's architecture. This was followed by a presentation of the system's FE and BE implementations with the exploration of an experiment conducted to create programming exercises using the GPT model and further the integration of GPT into the system. These Sections 5.1, 5.2, and 5.3 shed light on the chosen technologies, specific methodologies, and encountered challenges. To validate the system's effectiveness, two evaluations were conducted. The first assessed the UX of the Strapi UI to conclude unsatisfactory results, while the second analyzed the UX of the developed UI with the key functionalities and an overall positive assessment in the Sections 6.2 and 6.3.

At the beginning of the project, several objectives were outlined. The primary and most important one was the development of an intuitive UI with good usability. The evaluation of the developed UI showed significant improvements regarding Strapi. Overall, the respondents evaluated the system positively, with one of the major strong points mentioning the intuitiveness and simplicity of the system. Therefore, the objective of having an intuitive UI with good usability was achieved. However, it is important to mention that despite an overall positive evaluation, there is always room for improvement.

The system effectively incorporated the reuse of the Agni Student UI in an editable version for course creation with expositive and evaluative contents. Moreover, the system enables the reusability of courses, expositives, evaluatives, and questions, allowing teachers to selectively clone courses by choosing specific modules, lessons, expositives, and evaluatives. It also provides the flexibility to sequence content for better course alignment by time-based and exercise completeness conditions, offering integrated gamification opportunities. Further, the time-based sequencing can be individualized with delays for classes or students. These three mentioned objectives were successfully achieved.

As for system extensions, although the primary aim was to incorporate a variety of external repositories, only Authorkit was integrated. This integration is an interesting milestone as it's a feature nearly not found in other e-learning systems. The assisted exercise generation with the GPT model also succeeded. Despite its presented limitations with generating completely correct code with matching test cases, it is an interesting functionality that can assist teachers in creating programming exercises. An unfulfilled objective is the system's integration with an LMS, which would have simplified its use for teachers.

7.1 Contributions

The major contributions and findings of this thesis were:

- The system: One of the major contributions is the system developed, which can be used in classes and is available online[*]
- Published Paper: A paper was published with the title "Can a headless CMS provide a good UX to teachers?", detailing the evaluation of Strapi and the first approaches to the developed UI [18].
- Editable Agni Student UI: Using the student UI of Agni and making it editable was one of the key concepts of the UI. Showcasing the teacher's edits in real-time and how it affects what the students see was expected to be an intuitive concept. The evaluation and general positive consent with specific feedback describing the system as simple and intuitive approves this.

[*] https://agni.dcc.fc.up.pt/

- AuthorKit integration: The systems integration of an external exercises repository, namely AuthorKit, is a rarely found feature. Despite its rareness, it brings two systems together, making them improve from each other. These features should be implemented in more VLE to enhance the exercise landscape available to the teacher.

- Exercise generation with GPT: The experiment showcased limitations of the GPT model, which are the generation of valid JSON formats and difficulties in generating an exercise solution with correct test cases. Therefore, an alternative approach to pass a JSON schema within the parameters was presented, which was integrated into the system to assist teachers in generating programming exercises.

- Teacher as Focus: As explained in the first chapters, most research regarding e-learning focuses on students. This thesis instead highlights the development of the teachers UI with interesting features and evaluations to ensure good usability and insights into the user's perspectives. This enrichens a lacking research landscape.

7.2 Future Work

Despite the positive evaluation of the system, there is always room for improvement and more detailed analysis.

A major point for future work is integrating the system with an LMS. This would facilitate teachers' use in their universities, where most use LMSs. The integration would be done through Learning Tools Interoperability (LTI) following the roadmap provided by J. P. Leal, R. Queirós, P. Ferreirinha, and J. Swacha [55].

Further evaluation of the system with an UX evaluation tool would give more insights into the UX and possible other improvement opportunities. The evaluations conducted during the project were executed outside an actual classroom setting, focusing primarily on system usability. A more holistic assessment would entail deploying the system within real classroom environments. This would allow educators to delve deeper into its features and pinpoint areas for potential improvements. To get a better understanding of the UX of teachers, questionnaires such as AttrakDiff, UEQ, and SASSIS could be used. Subsequently, three evaluations at different times could be interesting to capture the evolution over time. The initial assessment would be conducted in the first or second week after the system's integration. This measures educators' first impressions and identifies immediate usability concerns as they set up courses and onboard students. A mid-semester review could offer insights into the system's performance during the semester, highlighting both its strengths and areas of contention. Lastly, an end-of-semester evaluation would reveal if sustained use addresses initial concerns or if new challenges arise with time.

Besides a more in-depth evaluation, some features could be enhanced or integrated to improve the system's capabilities further. Starting with integrating more external exercise repositories to broaden the resources available for educators. Gamification features could also significantly boost user engagement. For students, earning points, badges, or achieving levels can make learning more engaging. On the teacher side, a comparative analysis showcasing student progress against other classes from other teachers or historical data could motivate teachers to better their classes to beat their peers or last occurrences. A more diverse array of exercise types would be another possibility to enhance the system. The current offering could be augmented with assignments, open-ended questions, and mechanisms for more qualitative feedback. Given the described criticism about automated assessments, the system could allow teachers to pose questions during or post-submission. This can validate the understanding and prevent rote learning. While the platform is presently JS-centric, there's a clear need to accommodate exercises with HTML and CSS. Over time, adding languages such as Python and Java would make the system more complete and usable in a broader context. A communicative bridge between educators and learners can be further interesting. Features such as a chat system or Q&A forum, with options for public or private exchanges, can encourage a collaborative and supportive learning environment.

This thesis presented an experiment and the integration of the GPT model to assist teachers in creating programming exercises. To validate the chosen approach of passing a JSON schema in the parameters, the experiment of generating 120 exercises could be replicated using this approach. Therefore validating its use and finding possible limitations. Despite generating programming exercises with a given description of the teacher, teachers could select existing exercises and generate new ones based on the selected ones. Another possibility would be the generation of quizzes or a whole course. However, these language-based models can not only generate text. They can also analyze it and give suggestions. This could be leveraged as a fictive person or animal that, while the teacher creates his course, can analyze it and give feedback and improvement suggestions.

Appendix A

Agni Student User Interface

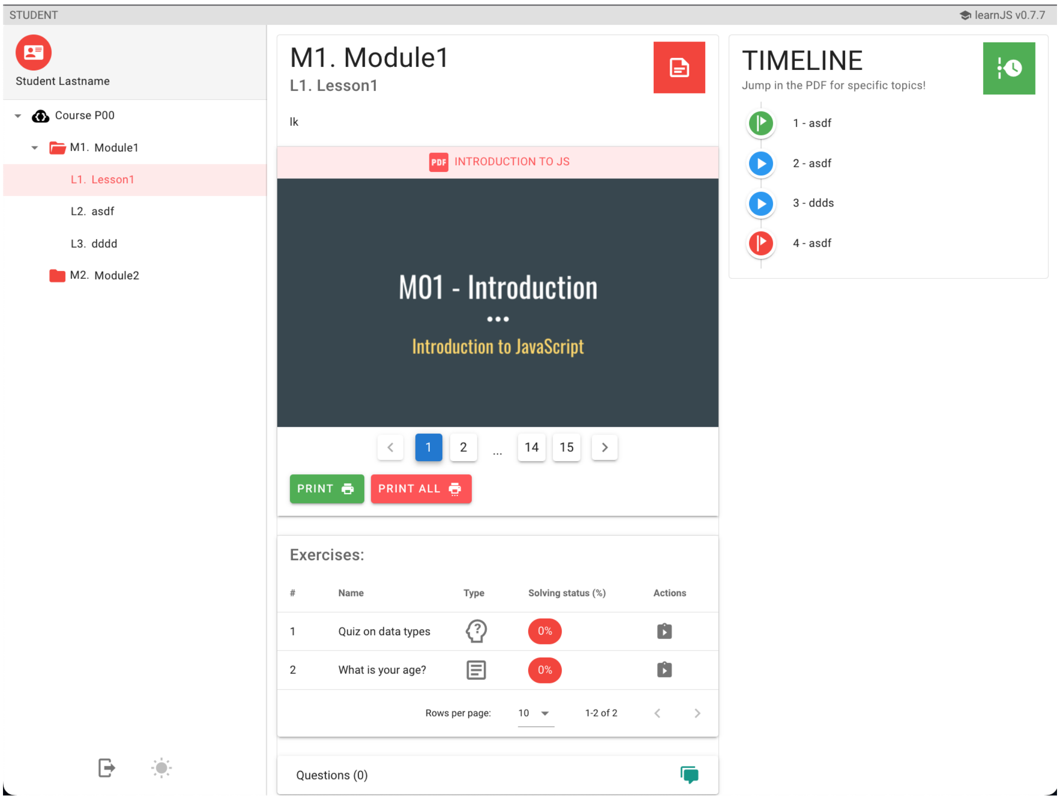

FIGURE A.1: Agni Student UI

Appendix B

Data Model

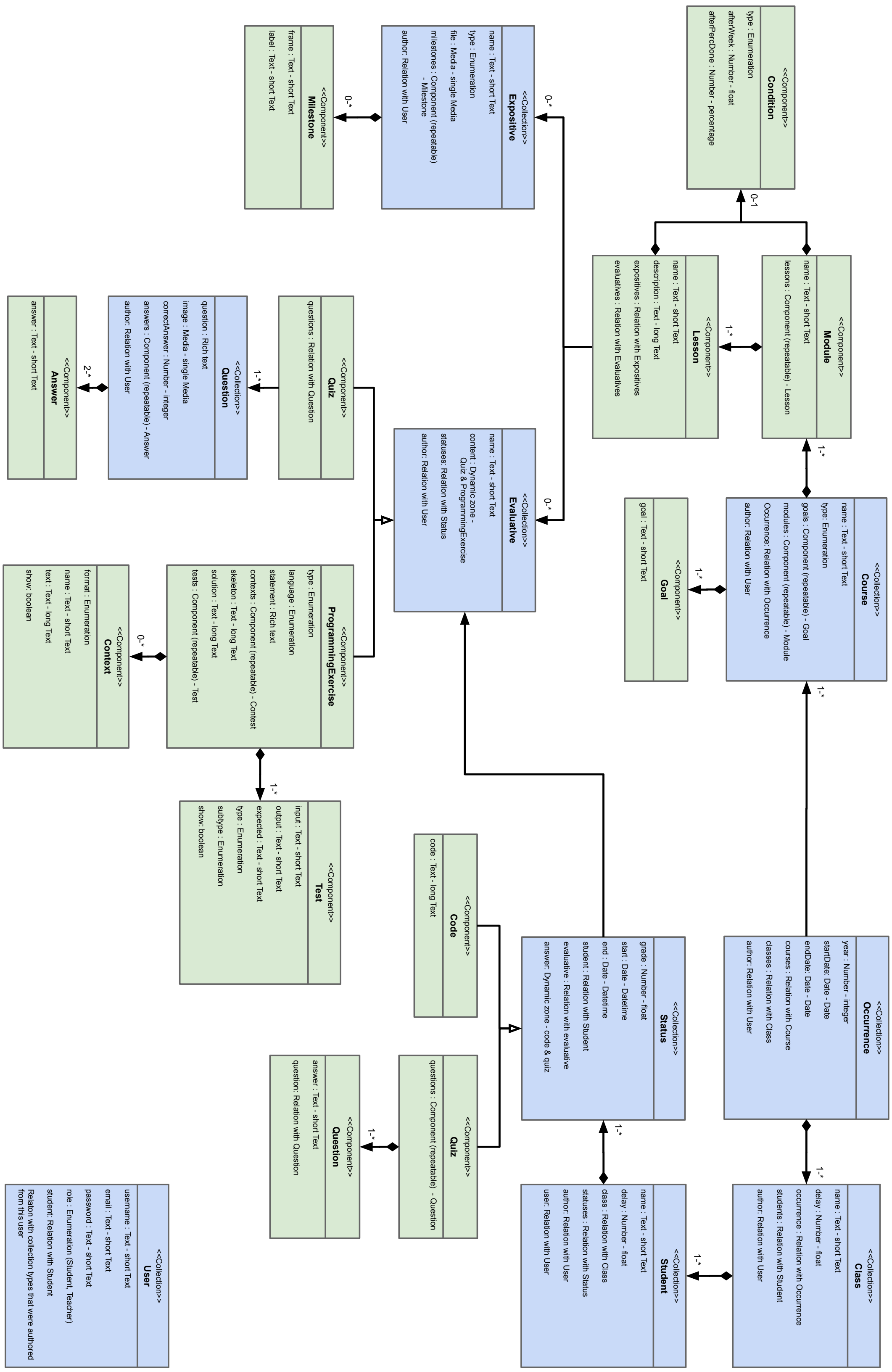

FIGURE B.1: Complete Data Model

Appendix C

Automated Exercise Generation Experiment

C.1 Prompts

First prompt used to generate programming exercises:

```
Create a programming exercise with the parameters: title,
difficulty, context, task, input description, output description,
example, solution language, solution code, and 5 input output
tests.
```

Final prompt to generate the 120 programming exercises:

```
Create a new programming exercise.  It has to be solvable in
various programming languages

Define:
title (in english, portugues, italian and polish),
difficulty ("beginner", "easy", "average", "hard", "master"),
context (you can make a detailed description which refers to a
real situation) (in english, portugues, italian and polish),
task (in english, portugues, italian and polish),
input description (in english, portugues, italian and polish),
output description (in english, portugues, italian and polish),
a basic example of input and output,
solution language,
solution code (which has to be a function without calling the
function with "input()"),
at least 5 tests with input output that cover the corner cases
(if the function has two or more parameters, the input should
look like:  input:"parameter1;parameter2")

write it in a json file in the following format:
{"title":  {"english":...,...}, "difficulty":  ...,
"context":  {"english":  ...,...}, "task":  {"english":
...,...}, "input_description":  {"english":  ...,...},
"output_description":  {"english":  ...,...}, "example":
{"input":  ...,"output":  ...}, "solution_language":  ...,
"solution_code":  ..., "tests":  [{"input":  ...,"output":
...},...]}
```

C.2 Final Prompt Responses

```
{
  "title": {
    "english":"Vowels Counter",
    "portugues":"Contador de Vogais",
    "italian":"Contatore di vocali",
    "polish":"Licznik samoglosek"
  },
  "difficulty":"beginner",
  "context": {
    "english":"You are asked to create a program that reads a text
      input and returns the number of vowels found in it.",
    ...
  },
  "task": {
    "english":"Create a function that receives a string and returns the
      number of vowels in it.",
    "portugues":"...",
    ...
  },
  "input_description": {
    "english":"A string of characters",
    "portugues":"Uma string de caracteres",
    ...
  },
  "output_description": {
    "english":"An integer with the count of vowels in the input string",
    "portugues":"...",
    ...
  },
  "example":{"input":"hello world","output":3},
  "solution_language":"Python",
  "solution_code":"def count_vowels(s):\n vowels = ['a', 'e', 'i', 'o', 'u']\n
    count = 0\n for c in s:\n if c.lower() in vowels:\n count += 1\n return count",
  "tests": [
    {"input":"the quick brown fox jumps over the lazy dog","output":11},
    {"input":"antidisestablishmentarianism","output":11},
    {"input":"aeiou","output":5},
    {"input":"rhythm","output":0},
    {"input":"Supercalifragilisticexpialidocious","output":16}
  ]
}
```

LISTING C.1: Correct generated Exercise

```
{
  "title": {
    "english": "Largest Word Finder",
    "portuguese": "Encontrador da Maior Palavra",
    "italian": "Trovatore della Parola Piu Lunga",
    "polish": "Znajdz Najdluzsze Slowo"
  },
  "difficulty": "easy",
  "context": {
    "english": "You are tasked to find the largest word in a given sentence.",
    ...
  },
  "task": {
    "english": "Create a function that receives a sentence and returns the largest
      word in it. If there are multiple words with the same length, return the
      first one in the sentence.",
    ...
  },
  "input_description": {
    "english": "A string representing a sentence.",
    ...
  },
  "output_description": {
    "english": "A string with the largest word in the sentence.",
    ...
  },
  "example": {
    "input": "The quick brown fox jumps over the lazy dog",
    "output": "quick"
  },
  "solution_language": "Python",
  "solution_code": "def largest_word(sentence):\n words = sentence.split()\n
    largest = ''\n for word in words:\n if len(word) > len(largest):\n
    largest = word\n return largest",
  "tests": [
    { "input": "Hello, world!", "output": "world" },
    {
      "input": "I am a developer. What is your profession?",
      "output": "profession?"
    },
    { "input": "Mathematics is not a spectator sport.", "output": "Mathematics" },
    ...
  ]
}
```

LISTING C.2: Code Error (first test case)

```
{
  "title":{
    "english":"FizzBuzz problem with a twist",
    "portugues":"Problema FizzBuzz com um toque especial",
    "italian":"Problema FizzBuzz con un tocco speciale",
    "polish":"Problemat FizzBuzz z niespodzianka"
  },
  "difficulty":"easy",
  "context":{
    "english":"FizzBuzz is a classic coding problem where you need to print numbers
      from 1 to n. If the number is divisible by 3, you should print 'Fizz'. If the
      number is divisible by 5, you should print 'Buzz'. If the number is divisible
      by both 3 and 5, you should print 'FizzBuzz'. Now, what if we add another
      condition? Let's say, if the number is a multiple of 7, you should print
      'Bang'. If the number is divisible by both 5 and 7, you should print
      'BuzzBang'. If the number is divisible by 3, 5, and 7, you should print
      'FizzBuzzBang'.",
    ...
  },
  "task":{
    "english":"Write a function that given n, prints the FizzBuzzBang output
      from 1 to n",
    ...
  },
  "input_description":{
    "english":"An integer n (1 <= n <= 10^6)",
    ...
  },
  "output_description":{
    "english":"Print the FizzBuzzBang output from 1 to n, each on a new line.",
    ...
  },
  "example":{ ... },
  "solution_language":"Python",
  "solution_code":"def fizz_buzz_bang(n):\n result = []\n for i in range(1, n+1):\n
    temp = ''\n if i % 3 == 0:\n temp += 'Fizz'\n if i % 5 == 0:\n temp += 'Buzz'\n
    if i % 7 == 0:\n temp += 'Bang'\n if not temp:\n temp = str(i)\n
    result.append(temp)\n return '\\n'.join(result)",
  "tests":[
    {"input":"15","output":"..."},
    {"input":"30","output":"..."},
    {"input":"1","output":"1"},
    {"input":"7","output":"1\n2\nFizz\n4\nBuzz\nFizz\nBang"}
```

LISTING C.3: JSON Error (missing brackets at the end)

C.3 Integrated Code Snipped

```
let prompt = "Create a JavaScript programming exercise with the description: "
    + data.description
const schema = {
  "type": "object",
  "properties": {
    "name": { "type": "string", "description": "Name of the exercise" },
    "statement": {
      "type": "string",
      "description": "Statement for the exercise"
    },
    "solution": {
      "type": "string",
      "description": "Solution code for the exercise in JavaScript"
    },
    "tests": {
      "type": "array",
      "description": "Input, output tests to verify the student's solution",
      "minItems": 3,
      "items": {
        "type": "object",
        "properties: {
          "input": { "type": "string", "description": "input for the test" },
          "expected": {
            "type": "string",
            "description": "expected output for the test"
          }
        },
        "required": ["input", "expected"]
      }
    }
  },
  "required": ["name", "statement", "solution", "tests"]
}
const chatCompletion = await openai.chat.completions.create({
  model: "gpt-3.5-turbo-0613",
  messages: [
    { role: "assistant", content: "I am a teacher to help you create JavaScript
        programming exercises" },
    { role: "user", content: prompt}],
  functions: [{ name: "set_exercise", parameters: schema }],
  function_call: {name: "set_exercise"}
})
```

LISTING C.4: Code to generate Programming Exercises by the Teacher

Appendix D

Strapi User Interface Evaluation

D.1 Evaluation Tasks

Task 1 Creation of a Course:

Create a course with 4 Modules.

The first Module should have 2 Lessons.

The first Lesson should contain a pdf file of your choice.

The second lesson should contain another pdf File of your choice and one Quiz, and 2 Programming exercises.

The Quiz should have 4 questions, each of them having 4 possible answers.

One of the Programming exercises should be a basic "Hello World!" exercise, and the second one the creation of a "sum" function with two parameters. You should create these exercises on your own and not reuse already created ones.

After that, choose one of the already created programming exercises and create a new programming exercise based on it, only changing the name.

Add this programming exercise to the second Lesson.

Now, clone the course that you created and change the "Hello World!" exercise so that the student has to return "Hello my friends!" without changing anything in the first created course with its "Hello World!" exercise.

Task 2 Creation of the students:

Create an Occurrence for the year 2023 with one Class and 5 Students.

Associate the Course that you created in task 1 to this occurrence.

(i) All information, such as names, descriptions, . . . are of your choice.

D.2 User Interface

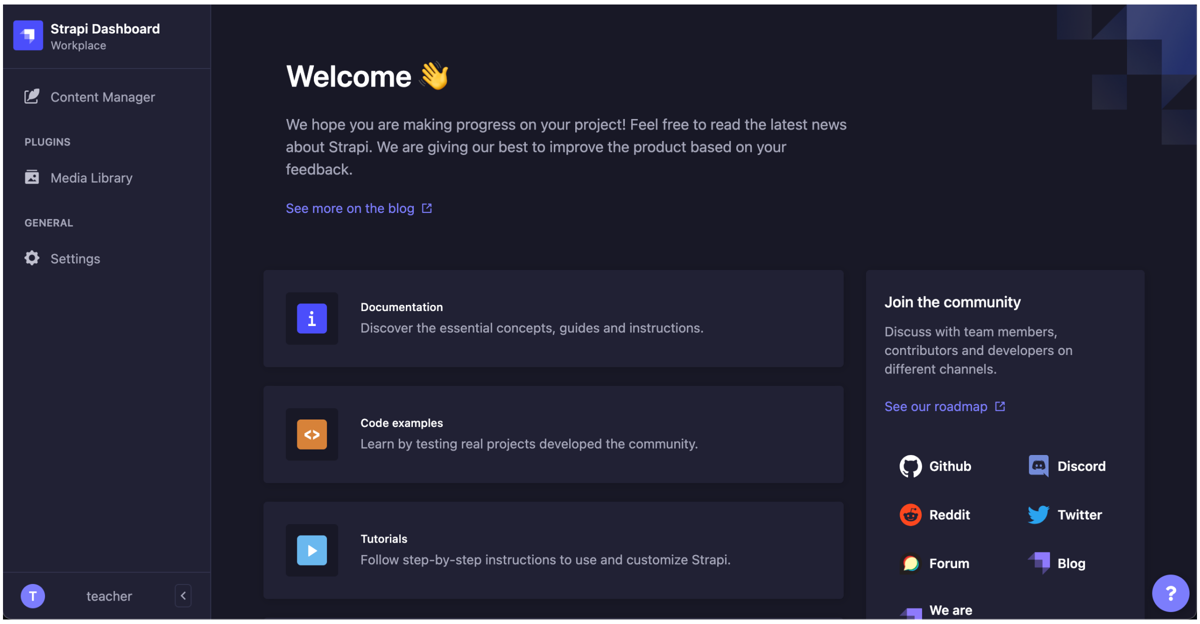

FIGURE D.1: Strapi Dashboard Interface

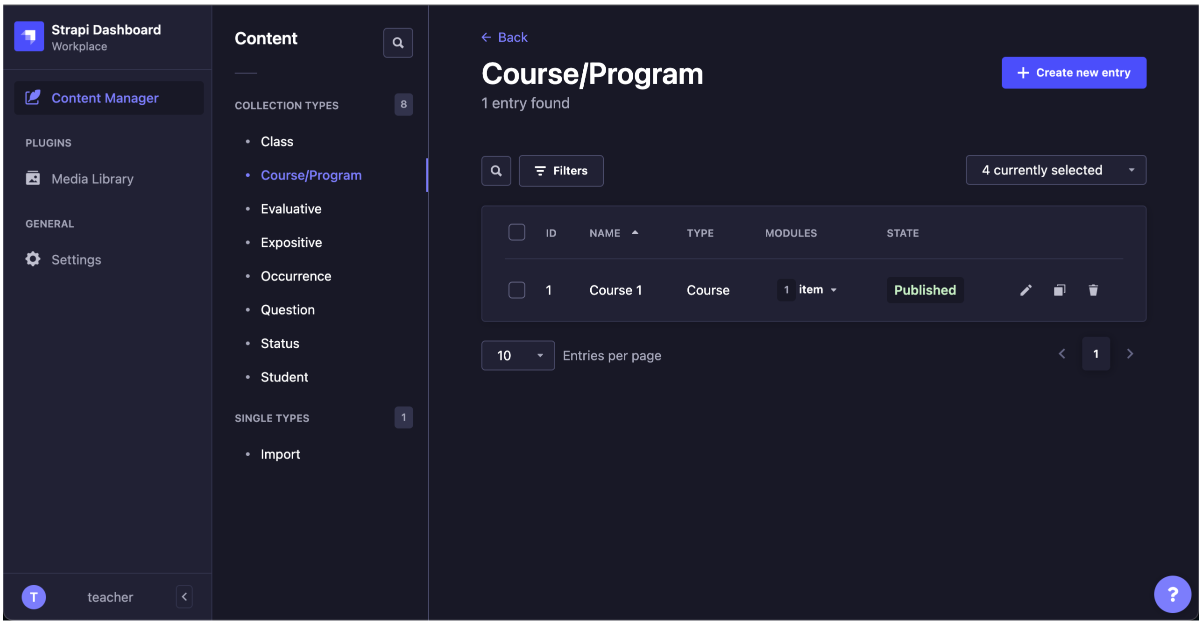

FIGURE D.2: Strapi Content Manager Interface

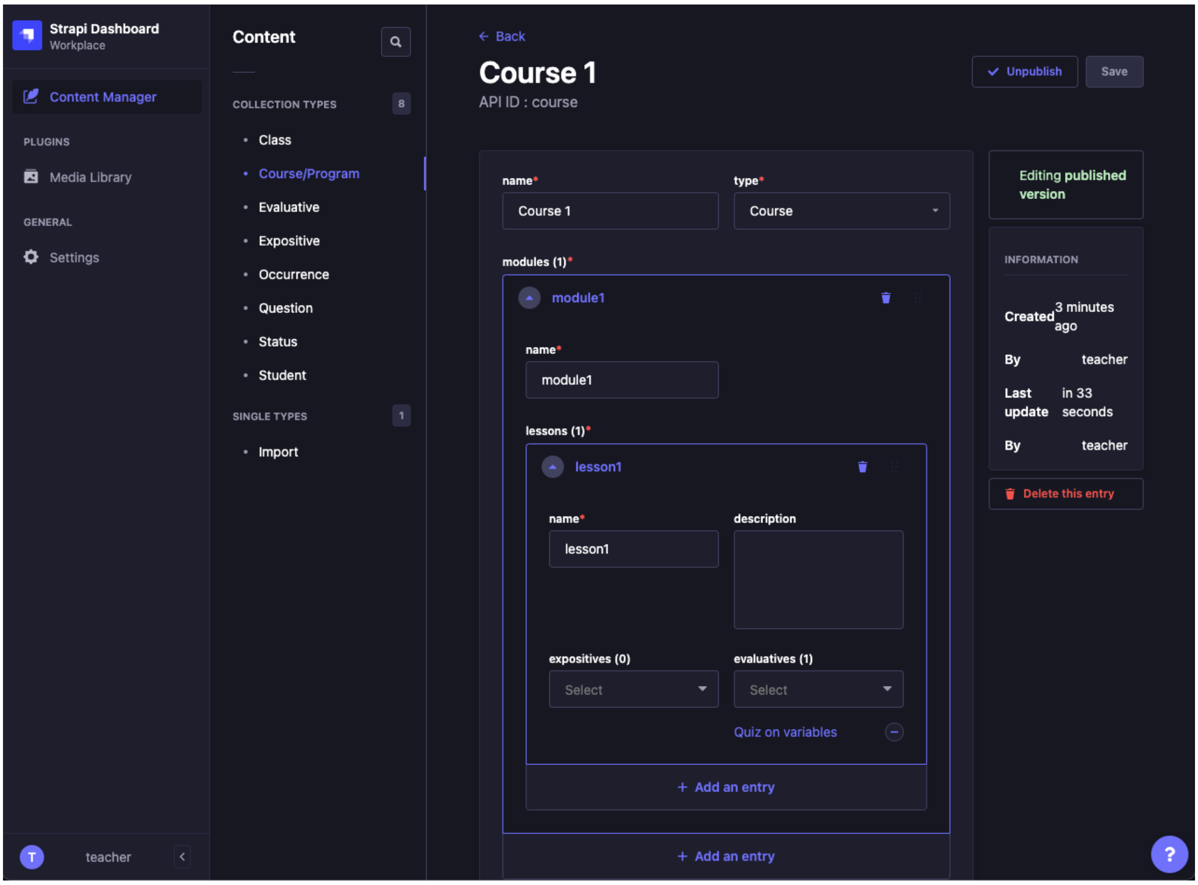

FIGURE D.3: Strapi Interface for creating a Course (Content Manager)

D.3 Questionnaire

Figure D.4 displays the questions within their heuristic category asked to the evaluators for the evaluation of the Strapi UI. The heuristics 1-13 had the possible choices of "Never", "Almost never", "Regular", "Almost always", "Always", and "Not applicable". For the classification, the possible options were "Very Good", "Good", "Sufficient", "Insufficient", and "Bad". Finally, the answers for the observations were in long text format.

1. Visibility: Visibility of system state (appropriate feedback in reasonable time)
   - A. When I ask the system for help, is the answer clear?
   - B. When I execute a task, does the system inform me about what is happening?
   - C. Are the buttons used to perform the most important tasks clearly identified?
   - D. Is the status of the buttons (selected/unselected) clearly indicated?
2. Compatibility: Compatibility between the system and the real world (language familiar to the user)
   - A. When I try to run a task, do I quickly find the desired button?
   - B. Is the order of the buttons in a familiar sequence?
   - C. When I press a button, is the result expected?
   - D. Do the commands grouped in a menu all belong to the category indicated by the button text of that menu?
3. Freedom: User freedom and control (it must be possible for the user to undo or redo operations)
   - A. When I make a mistake, does the system allow me to go back?
   - B. Can I interrupt an action and resume it at any time?
   - C. Can I cancel an operation that is in progress?
   - D. Can I delete any changes that were made and go back to the previous state?
4. Consistency: Consistency and standards (the system must be clearly understood and follow known conventions)
   - A. Is the location of buttons and windows maintained when I switch screens?
   - B. Do the buttons always have the same meaning when I switch the screen?
   - C. Is the meaning of the color code consistent?
   - D. Is it possible to use scroll in all windows?
5. Prevention: Error prevention (avoid them)
   - A. Does the system notify me when data entry problems occur, before I run validation?
   - B. When I do a validation, does the system give an error message if the data format is not what I expected?
   - C. Am I warned by the system if I am about to make a serious mistake?
   - D. Is there a clear separation between the buttons that allow the occurrence of serious errors and the remaining buttons?
6. Emphasis: Emphasis on recognition (minimize user memory effort by making objects, actions and options always present)
   - A. Do the colors used in the texts conform to accepted conventions for their meaning?
   - B. Does the text contained in each button convey the idea of what is expected?
   - C. Is the information on the screen available where I expect it?
   - D. Are the items grouped by gender in different logical zones?
7. Flexibility: Flexibility and efficiency in use (the user should be allowed to customize or schedule frequent actions - e.g. create shortcuts)
   - A. Can I configure the screen?
   - B. Are there shortcut keys to perform commonly used functions?
   - C. Can I temporarily disable some functions?
8. Aesthetics: Aesthetics and minimalist design (only essential information)
   - A. Is the information on the screen just what I need?
   - B. Does the information on the screen stand out from the background?
   - C. Aesthetically, is the system pleasant in terms of: colors, brightness, etc.?
9. Help to Users: Helps users recognize, diagnose and recover from errors (error messages must be clear and suggest solutions)
   - A. Are the error/help messages clear and appropriate?
   - B. Do the error messages accurately indicate the problem?
   - C. Are the messages short and to the point?
10. Help and documentation: Help and documentation (documentation always available)
    - A. Can I easily search for information?
    - B. Is the help function clearly visible?
    - C. Is the information accurate, complete and understandable?
11. Easiness: Easiness of learning
    - A. Is the system intuitive (I understand it easily)?
    - B. Do I find it easy to learn to work with the system?
12. Speed: System response speed
    - A. Is the response time for the operations carried out sufficiently short?
    - B. Is the response time on switching tasks short enough (Example: you are seeing "pending" and change it to "impressions") (ex: está a ver "pendentes" e altera para "impressões") ?
13. Reliability: Reliability of your functions (how often do functions work without issues?)
    - A. Creating a Course with Modules and Lessons?
    - B. Creating expositive content?
    - C. Creating a Programming Exercise?
    - D. Creating a Quiz?
    - E. Creating Questions?
    - F. Associating Questions to a Quiz?
    - G. Associating expositive or evaluative content to a Lesson?
    - H. Creating a programming exercise based on an existing one?
    - I. Cloning an existing course with a small change in one exercise?
    - J. Creating an Occurrence, Class or Students?
    - K. Associating a Course to an Occurrence or Students to a Course?
14. Classification
    - A. Given all the parameters you analysed how would you classify Strapi?
15. Observations: Strong points and weak points
    - A. Strong points?
    - B. Weak points?
    - B. Improvement suggestions?

FIGURE D.4: Strapi User Interface Evaluation Questionnaire

D.4 Results

	S1	S2	S3	S4	S5	%	Mean
1.						80	3.25
A	4			2		40	3.00
B	3	2	4	3	5	100	3.40
C	3	2	2	3	4	100	2.80
D	5	4	2	4		80	3.75
2.						90	3.17
A	3	2	3	3	5	100	3.20
B	4	2	2	1		80	2.25
C	5	3	3	3	5	100	3.80
D	4	3	3	3		80	3.25
3.						70	3.14
A	4	4	4	3		80	3.75
B	3	2	3	1		80	2.25
C	5		3			40	4.00
D	3	2	3	4		80	3.00
4.						80	3.94
A	4	2	3	5		80	3.50
B	5	3	5	4		80	4.25
C	5	4	4	4		80	4.25
D	5	2	4	4		80	3.75
5.						45	3.11
A		4	4	4		60	4.00
B		4	2	3		60	3.00
C		3	2			40	2.50
D			2			20	2.00
6.						70	3.93
A	5	3	3	5		80	4.00
B	4	4	4	5		80	4.25
C	5	3	3	4		80	3.75
D	4		3			40	3.50
7.						67	2.00
A	4	1	3	3		80	2.75
B		1	2	1		60	1.33
C	3	1		1		60	1.67

	S1	S2	S3	S4	S5	%	Mean
8.						80	3.25
A	4	3	2	2		80	2.75
B	4	3	4	3		80	3.50
C	4	3	3	4		80	3.50
9.						67	3.20
A		2	2	4		60	2.67
B	4	3		3		60	3.33
C	4	3	4	3		80	3.50
10.						53	3.25
A	4			3	2	60	3.00
B	4		5	2		60	3.67
C	4		2			40	3.00
11.						80	2.25
A	4	1	2	1		80	2.00
B	4	2	3	1		80	2.50
12.						50	4.20
A	5	3	5	4		80	4.25
B	4					20	4.00
13.						100	2.91
A	2	2	5	2	5	100	3.20
B	2	2	3	2	5	100	2.80
C	2	1	2	1	4	100	2.00
D	2	3	2	1	5	100	2.60
E	3	2	2	2	5	100	2.80
F	1	4	3	3	5	100	3.20
G	3	1	4	3	5	100	3.20
H	2	4	4	2	5	100	3.40
I	2	3	3	1	5	100	2.80
J	2	3	4	2	5	100	3.20
K	3	2	2	2	5	100	2.80
14.							
A	3	2	3	2	4	100	2.80

Strong points	
S1	General easy to use and good UI
S2	Fast and responsive
S3	Easy creation of course, modules, and lessons
S4	Good general UI
S5	Easy creation of course, modules, and lessons
Weak points	
S1	Not all of the specifications are clearly defined
S2	Bad abstraction and separation of concepts, lack of shortcuts. Having to create exercises and expositives separately of a lesson
S3	Poor collection types structure (for instance, the course should be bigger). Poor ID association. Having to create exercises and expositives separately of a lesson
S4	Confusing
S5	Difficult to search for information/help
Improvement suggestions	
S1	Tutorial to explain the UI
S2	Resolve weak points
S3	Resolve weak points
S4	Tutorial to explain the UI
S5	/

TABLE D.1: Strapi Evaluation Detailed Results

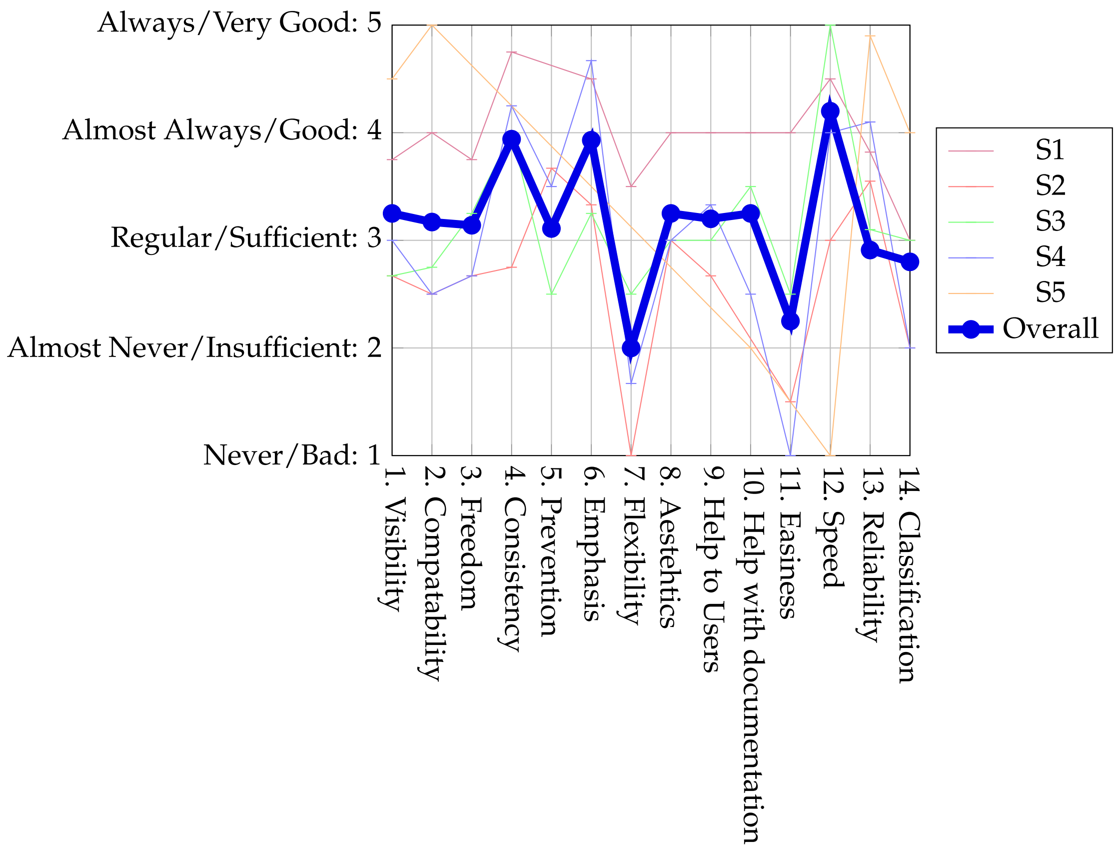

FIGURE D.5: Strapi UI Evaluation Results (line chart)

Appendix E

Final User Interface Evaluation

E.1 User Interface

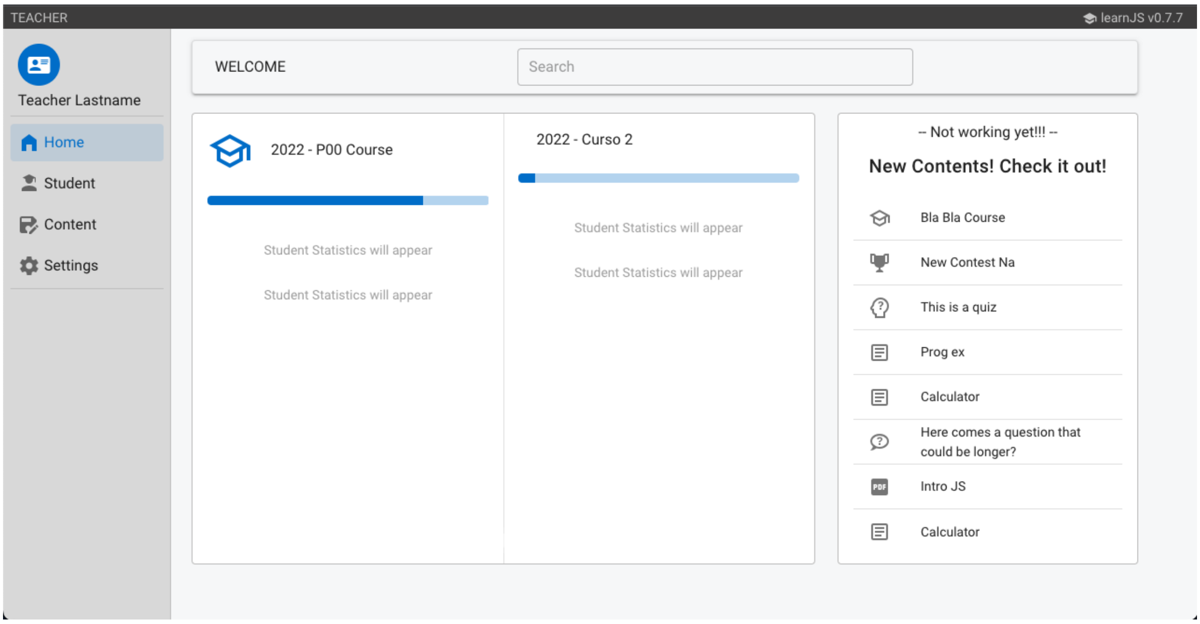

FIGURE E.1: Teacher Interface for the Home Menu (evaluated UI)

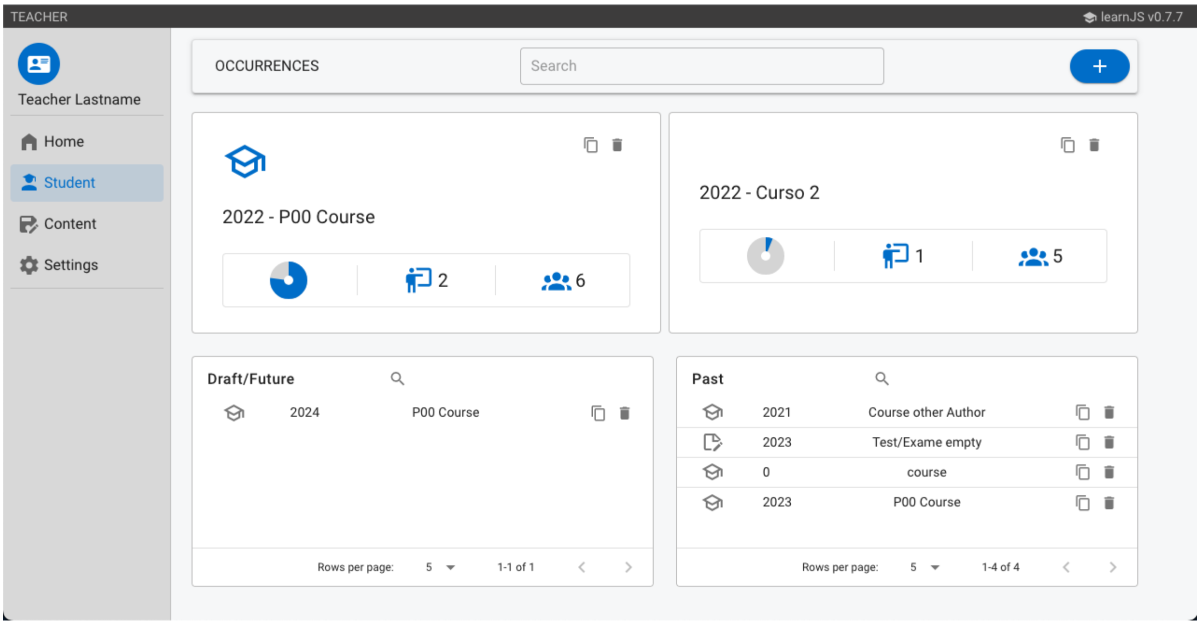

FIGURE E.2: Teacher Interface to manage Students (evaluated UI)

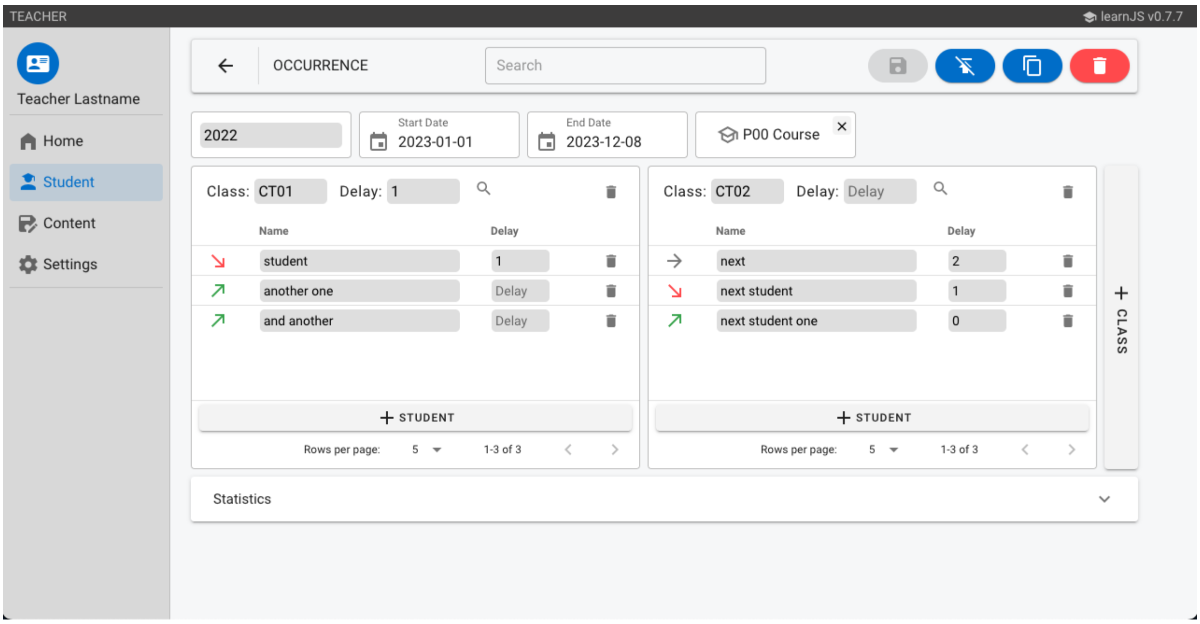

FIGURE E.3: Teacher Interface to create and edit Occurrence (evaluated UI)

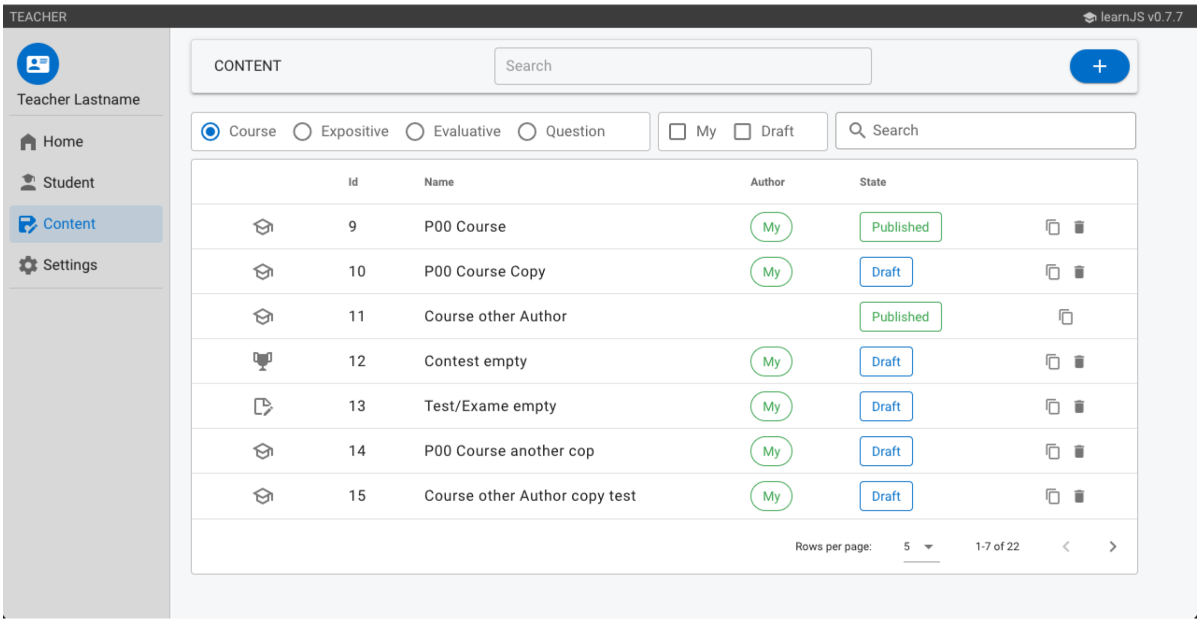

FIGURE E.4: Teacher Interface to manage Content (evaluated UI)

E.2 Questionnaire

1. Visibility: Visibility of system state (appropriate feedback in reasonable time)
   A. When I ask the system for help, is the answer clear?
   B. When I execute a task, does the system inform me about what is happening?
   C. Are the buttons used to perform the most important tasks clearly identified?
   D. Is the status of the buttons (selected/unselected) clearly indicated?
2. Compatibility: Compatibility between the system and the real world (language familiar to the user)
   A. When I try to run a task, do I quickly find the desired button?
   B. Is the order of the buttons in a familiar sequence?
   C. When I press a button, is the result expected?
   D. Do the commands grouped in a menu all belong to the category indicated by the button text of that menu?
3. Freedom: User freedom and control (it must be possible for the user to undo or redo operations)
   A. When I make a mistake, does the system allow me to go back?
   B. Can I interrupt an action and resume it at any time?
   C. Can I cancel an operation that is in progress?
   D. Can I delete any changes that were made and go back to the previous state?
4. Consistency: Consistency and standards (the system must be clearly understood and follow known conventions)
   A. Is the location of buttons and windows maintained when I switch screens?
   B. Do the buttons always have the same meaning when I switch the screen?
   C. Is the meaning of the color code consistent?
   D. Is it possible to use scroll in all windows?
5. Prevention: Error prevention (avoid them)
   A. Does the system notify me when data entry problems occur, before I run validation?
   B. When I do a validation, does the system give an error message if the data format is not what I expected?
   C. Am I warned by the system if I am about to make a serious mistake?
   D. Is there a clear separation between the buttons that allow the occurrence of serious errors and the remaining buttons?
6. Emphasis: Emphasis on recognition (minimize user memory effort by making objects, actions and options always present)
   A. Do the colors used in the texts conform to accepted conventions for their meaning?
   B. Does the text contained in each button convey the idea of what is expected?
   C. Is the information on the screen available where I expect it?
   D. Are the items grouped by gender in different logical zones?
7. Flexibility: Flexibility and efficiency in use (the user should be allowed to customize or schedule frequent actions - e.g. create shortcuts)
   A. Can I configure the screen?
   B. Are there shortcut keys to perform commonly used functions?
   C. Can I temporarily disable some functions?
8. Aesthetics: Aesthetics and minimalist design (only essential information)
   A. Is the information on the screen just what I need?
   B. Does the information on the screen stand out from the background?
   C. Aesthetically, is the system pleasant in terms of: colors, brightness, etc.?
9. Help to Users: Helps users recognize, diagnose and recover from errors (error messages must be clear and suggest solutions)
   A. Are the error/help messages clear and appropriate?
   B. Do the error messages accurately indicate the problem?
   C. Are the messages short and to the point?
10. Help and documentation: Help and documentation (documentation always available)
    A. Can I easily search for information?
    B. Is the help function clearly visible?
    C. Is the information accurate, complete and understandable?
11. Easiness: Easiness of learning
    A. Is the system intuitive (I understand it easily)?
    B. Do I find it easy to learn to work with the system?
12. Speed: System response speed
    A. Is the response time for the operations carried out sufficiently short?
    B. Is the response time on switching tasks short enough (Example: you are seeing "pending" and change it to "impressions") (ex: está a ver "pendentes" e altera para "impressões") ?
13. Reliability: Reliability of your functions (how often do functions work without issues?)
    A. Creating a Course with Modules and Lessons?
    B. Creating expositive content?
    C. Creating a Programming Exercise?
    D. Creating a Quiz?
    E. Creating Questions?
    F. Adding existing exercises to a Course?
    G. Creating a programming exercise based on an existing one?
    H. Cloning an existing course and making small adjustments??
    I. Creating an Occurrence, Class or Students?
14. Classification
    A. Given all the parameters you analysed how would you classify the Agni Teacher UI?
15. Observations: Strong points and weak points
    A. Strong points?
    B. Weak points?
    B. Improvement suggestions?

FIGURE E.5: Final User Interface Evaluation Questionnaire

Figure E.5 displays the questions within their heuristic category asked to the evaluators for the evaluation of the developed UI. The heuristics 1-13 had the possible choices of "Never", "Almost never", "Regular", "Almost always", "Always", and "Not applicable". For the classification, the possible options were "Very Good", "Good", "Sufficient", "Insufficient", and "Bad". Finally, the answers for the observations were in long text format.

All categories, despite reliability with some small adjustments, are the same questions as the evaluation of Strapis UI displayed in D.

E.3 Results

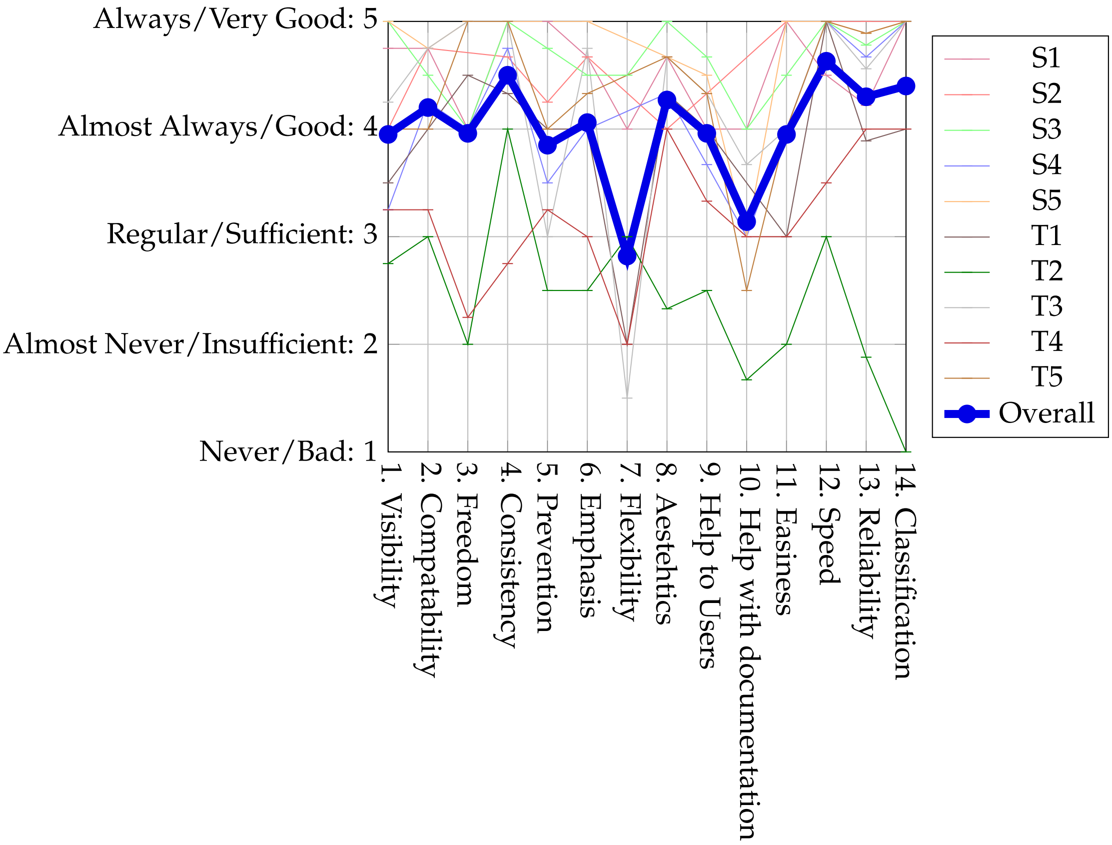

FIGURE E.6: Final UI Evaluation Results (line chart)

	S1	S2	S3	S4	S5	T1	T2	T3	T4	T5	%	Mean
1.											93	3.95
A	5		5	2		4	3	4	3	4	80	3.75
B	5		5	2	5	3	2	4	3	3	90	3.56
C	4	4	5	4	5	3	2	4	4	4	100	3.90
D	5	4	5	5	5	4	4	5	3	5	100	4.50
2.											100	4.20
A	5	5	4	3	5	3	3	4	3	3	100	3.80
B	4	5	5	5	5	4	2	5	3	5	100	4.30
C	5	5	4	4	4	4	3	5	4	4	100	4.20
D	5	4	5	5	5	5	4	5	3	4	100	4.50
3.											60	3.96
A	3		4	5	5	3	2	5	2	5	90	3.78
B	5		4	3		5		5	2		60	4.00
C	4					5			2		30	3.67
D	4		4		5	5			3	5	60	4.33
4.											80	4.50
A	5	4	5	5	5	4	3	5	3	5	100	4.40
B	5	5	5	4		4	4	5	3	5	90	4.44
C	5	5	5	5		5	5	5	3	5	90	4.78
D	5			5	5				2		40	4.25
5.											65	3.85
A		5	4	4		4	2	3	3	4	80	3.63
B		5	5	3		4		3	4		60	4.00
C	5	4	5			4		3	3		60	4.00
D	5	3	5			4	3		3		60	3.83
6.											85	4.06
A	5		5	5		5	2	5	3	5	80	4.38
B	5	4	4	4	5	4	4	5	3	5	100	4.30
C	4	5	4	3	5	3	2	4	3	3	100	3.60
D		5	5			4	2	5	3		60	4.00
7.											37	2.82
A	4		4			2	3	2	2		60	2.83
B	4		5					1	2		40	3.00
C									2		10	2.00
8.											100	4.27
A	4	4	5	3	5	3	2	4	4	4	100	3.80
B	5	4	5	5	5	5	3	5	4	5	100	4.60
C	5	4	5	5	4	5	2	5	4	5	100	4.40
9.											90	3.96
A	4	4	4	3	5	4	2	4	3	4	100	3.70
B		4	5	3		4		4	3	4	70	3.86
C	4	5	5	4	4	4	3	5	4	5	100	4.30
10.											73	3.14
A	4		4	3			2	3	3		60	3.17
B	4		3	2	1		1	4	3	1	80	2.38
C	4		5	4	5		2	4	3	4	80	3.88
11.											100	3.95
A	5	5	4	4	5	3	2	4	3	4	100	3.90
B	5	5	5	4	5	3	2	4	3	4	100	4.00
12.											80	4.63
A	4	5	5	5	5	5	3	5	4	5	100	4.60
B	5	5	5	5		5			3		60	4.67
13.											99	4.30
A	4	5	5	5	5	3	1	5	5	5	100	4.30
B	4	5	5	5	4	4	1	5	5	5	100	4.30
C	5	5	4	5	5	4	2	5	4	4	100	4.30
D	5	5	5	5	5	4	2	5	4	5	100	4.50
E	5	5	5	5	5	4	2	5	4	5	100	4.50
F	4	5	4	4	5	4		5	4	5	90	4.44
G	3	5	5	4	5	4	1	3	4	5	100	3.90
H	3	5	5	4	5	4	2	3	3	5	100	3.90
I	5	5	5	5	5	4	4	5	3	5	100	4.60
14.												
A	5	5	5	5	5	4	1	5	4	5	100	4.40

TABLE E.1: Final Evaluation Detailed Results (categories)

Strong points	
S1	Beautiful and intuitive UI
S2	Very easy to use and very intuitive, fast operation and color scheme are very neutral, so probably easy to work with for extended periods of time.
S3	Beautiful and simple UI
S4	Good and intuitive UI
S5	Really easy to understand
T1	The process of creating activities is easy and intuitive (quizzes, exercises, etc.)
T2	The tool.
T3	The plataform is intuitive and easy to use
T4	Simple and intuitive interface. It has all principal functionalities.
T5	Maintains the usability philosophy.
Weak points	
S1	What is in the improvements
S2	Did not notice undo functionality. Did not notice customization. Lack of confirmation on deletes and such improves speed but might lead to unwanted errors.
S3	Saving contents is not automated
S4	It can get tricky for a first-timer to add exercises to a class.
S5	I didn't find a help button, but I also didn't need one
T1	Difficulties to find buttons to create modules and lessons. unintuitive interface in this function. Missing logout and customized user data
T2	Complexity of the system - It is not easy to manage. It is hard to follow the goal and needs many clicks to do something.
T3	I was not able to use to clone an existing course and making small adjustments
T4	Some difficulties with first time usage. Sometimes I felt a little lost with so many windows. I don't know if it has to do with the color enhancement of the resource that is selected or it is simply a matter of excess information (visual pollution). Error when saving the course. Some errors when I select a programming exercise done by others (changing the type, adding tests, etc. The exercise selection window should have more context, I suggest <course name>/<exercise name>. Answers in quizzes allow you to select multiple answers. Most of the time this is not the desirable behavior (switching to radiobutton)
T5	It appears to replicate an existing system (Moodle)
Improvement suggestions	
S1	Possiblity to clone exercises also within a course. Everywhere you can click, make the cursor to pointer (occurrence table,...)
S2	Include delete confirmation as an option. Improve customizability and hotkeys.
S3	Improve weak points
S4	Add a '+' (plus sign) into the menu after lessons so that adding exercises is easier.
S5	/
T1	Improve the more macro process of creating courses-modules-lessons. Help available at these stages
T2	Create all tasks in one step form (e.g., tell the number of courses, modules, and lessons and do them all simultaneously and not one by one) Create an onboarding experience. Change the color. It is too monotone and old school. You must have the option to select all options. Reduce the number of clicks to do something. Allow CSV upload with the data. Auto-save is a must-have. You create many interfaces simultaneously without validation. You must divide the interface into parts and create a focus group.
T3	Platform must be tested if there are errors (selection of dates in the past)
T4	Dynamic tutorial (facilitate onboarding). Gamification (PBL, lock/unlock levels through conditions, avatars, etc.)
T5	Do more of these tests and refinements

TABLE E.2: Final Evaluation Detailed Results (observations)

Bibliography

[1] L. Corporation, "Annual report for 2023," LinkedIn Corporation, Mountain View, CA, Tech. Rep., 2023. [Online]. Available: https://www.linkedin.com/business/talent/blog/talent-strategy/linkedin-most-in-demand-hard-and-soft-skills [Cited on page 1.]

[2] B. Xia, "An in-depth analysis of learning goals in higher education: Evidence from the programming education," *Journal of Learning Design*, vol. 10, p. 25, 03 2017. [Cited on page 1.]

[3] M. Piteira and C. Costa, "Learning computer programming: Study of difficulties in learning programming," in *Proceedings of the 2013 International Conference on Information Systems and Design of Communication*, ser. ISDOC '13. New York, NY, USA: Association for Computing Machinery, 2013, p. 75–80. [Online]. Available: https://doi.org/10.1145/2503859.2503871

[4] J. R. Anthony Robins and N. Rountree, "Learning and teaching programming: A review and discussion," *Computer Science Education*, vol. 13, no. 2, pp. 137–172, 2003. [Online]. Available: https://doi.org/10.1076/csed.13.2.137.14200

[5] S. Derus and A. M. Ali, "Difficulties in learning programming: Views of students," in *1st International Conference on Current Issues in Education (ICCIE 2012)*, 2012, pp. 74–79. [Cited on page 1.]

[6] Y. Bosse and M. A. Gerosa, "Why is programming so difficult to learn? patterns of difficulties related to programming learning mid-stage," *ACM SIGSOFT Software Engineering Notes*, vol. 41, no. 6, pp. 1–6, 2017.

[7] T. Jenkins, "On the difficulty of learning to program," in *Proceedings of the 3rd Annual Conference of the LTSN Centre for Information and Computer Sciences*, vol. 4, no. 2002. Citeseer, 2002, pp. 53–58. [Cited on page 1.]

[8] L. E. Winslow, "Programming pedagogy—a psychological overview," *SIGCSE Bull.*, vol. 28, no. 3, p. 17–22, sep 1996. [Online]. Available: https://doi.org/10.1145/234867.234872 [Cited on page 1.]

[9] R. Sabarinath and C. Quek, "A case study investigating programming students' peer review of codes and their perceptions of the online learning environment," *Education and Information Technologies*, vol. 25, 09 2020. [Cited on pages 1 and 16.]

[10] P. Denny, D. Cukierman, and J. Bhaskar, "Measuring the effect of inventing practice exercises on learning in an introductory programming course," in *Proceedings of the 15th Koli Calling Conference on Computing Education Research*, ser. Koli Calling '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 13–22. [Online]. Available: https://doi.org/10.1145/2828959.2828967

[11] K. von Hausswolff, "Practical thinking in programming education," in *Proceedings of the 17th Koli Calling International Conference on Computing Education Research*, ser. Koli Calling '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 203–204. [Online]. Available: https://doi.org/10.1145/3141880.3143780

[12] R. Millar *et al.*, "The role of practical work in the teaching and learning of science," *Commissioned paper-Committee on High School Science Laboratories: Role and Vision. Washington DC: National Academy of Sciences*, vol. 308, 2004. [Cited on page 1.]

[13] M. Kordaki, "A drawing and multi-representational computer environment for beginners' learning of programming using c: Design and pilot formative evaluation," *Computers & Education*, vol. 54, no. 1, pp. 69–87, 2010. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0360131509001845 [Cited on pages 1 and 16.]

[14] E. Verdú, L. M. Regueras, M. J. Verdú, J. P. Leal, J. P. de Castro, and R. Queirós, "A distributed system for learning programming on-line," *Computers & Education*, vol. 58, no. 1, pp. 1–10, 2012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S036013151100193X [Cited on page 10.]

[15] D. Muñoz de la Peña, F. Gómez-Estern, and S. Dormido, "A new internet tool for automatic evaluation in control systems and programming," *Computers & Education*, vol. 59, no. 2, pp. 535–550, 2012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0360131511003320 [Cited on pages 1, 11, and 16.]

[16] I. Maslov, S. Nikou, and P. Hansen, "Exploring user experience of learning management system," *The International Journal of Information and Learning Technology*, vol. ahead-of-print, 07 2021. [Cited on pages 1, 16, and 17.]

[17] L. Langreo, "What teachers really think about their learning management systems," Sep 2022. [Online]. Available: https://www.edweek.org/technology/what-teachers-really-think-about-their-learning-management-systems/2022/09 [Cited on pages 1, 16, and 17.]

[18] Y. Bauer, J. P. Leal, and R. Queirós, "Can a Content Management System Provide a Good User Experience to Teachers?" in *4th International Computer Programming Education Conference (ICPEC 2023)*, ser. Open Access Series in Informatics (OASIcs), R. A. Peixoto de Queirós and M. P. Teixeira Pinto, Eds., vol. 112. Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023, pp. 4:1–4:8. [Online]. Available: https://drops.dagstuhl.de/opus/volltexte/2023/18500 [Cited on pages 3 and 62.]

[19] M. Hassenzahl and N. Tractinsky, "User experience - a research agenda," *Behaviour & Information Technology*, vol. 25, no. 2, pp. 91–97, 2006. [Online]. Available: https://doi.org/10.1080/01449290500330331 [Cited on page 6.]

[20] V. Roto and E. Kaasinen, "The second international workshop on mobile internet user experience," in *Proceedings of the 10th international conference on Human computer interaction with mobile devices and services*, 09 2008, pp. 571–573. [Cited on page 6.]

[21] P. Desmet and P. Hekkert, "Framework of product experience," *International Journal of Design*, vol. 1, pp. 57–66, 04 2007. [Cited on page 6.]

[22] M. Hassenzahl, *The Thing and I: Understanding the Relationship Between User and Product*. Springer, 01 2005, vol. 3, pp. 31–42. [Cited on page 6.]

[23] N. Tractinsky, A. Katz, and D. Ikar, "What is beautiful is usable," *Interacting with Computers*, vol. 13, pp. 127–145, 12 2000. [Cited on page 6.]

[24] P. W. Jordan, *Designing pleasurable products: An introduction to the new human factors*. CRC press, 2000. [Cited on page 6.]

[25] K. BATTARBEE and I. KOSKINEN, "Co-experience: user experience as interaction," *CoDesign*, vol. 1, no. 1, pp. 5–18, 2005. [Online]. Available: https://doi.org/10.1080/15710880412331289917 [Cited on page 6.]

[26] E. Karapanos, J. Zimmerman, J. Forlizzi, and J.-B. Martens, "User experience over time: An initial framework," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '09. New York, NY, USA: Association for Computing Machinery, 2009, p. 729–738. [Online]. Available: https://doi.org/10.1145/1518701.1518814 [Cited on page 6.]

[27] A. H. Allam, A. R. C. Hussin, and H. M. Dahlan, "User experience: challenges and opportunities," in *Journal of Information Systems Research and Innovation 2013*, 2013. [Cited on page 7.]

[28] D. Benyon, *Designing User Experience*. Pearson Educación, 2019. [Online]. Available: https://books.google.pt/books?id=MXqFDwAAQBAJ [Cited on pages 7, 8, and 17.]

[29] M. Gualtieri, "Best practices in user experience (ux) design," *Design compelling user experiences to wow your customers*, pp. 1–17, 2009. [Cited on page 7.]

[30] A. Williams, "User-centered design, activity-centered design, and goal-directed design: A review of three methods for designing web applications," in *Proceedings of the 27th ACM International Conference on Design of Communication*, ser. SIGDOC '09. New York, NY, USA: Association for Computing Machinery, 2009, p. 1–8. [Online]. Available: https://doi.org/10.1145/1621995.1621997 [Cited on page 7.]

[31] D. A. Norman, "Human-centered design considered harmful," *Interactions*, vol. 12, no. 4, p. 14–19, jul 2005. [Online]. Available: https://doi.org/10.1145/1070960.1070976

[32] E. Schweikardt, "Sustainably oursuser centered is off center," *Interactions*, vol. 16, no. 3, p. 12–15, may 2009. [Online]. Available: https://doi.org/10.1145/1516016.1516019 [Cited on page 7.]

[33] S. H. Hsu, M. C. Chuang, and C. C. Chang, "A semantic differential study of designers' and users' product form perception," *International Journal of Industrial Ergonomics*, vol. 25, no. 4, pp. 375–391, 2000. [Online]. Available:

https://www.sciencedirect.com/science/article/pii/S0169814199000268 [Cited on pages 8 and 17.]

[34] D. Bartram, "The perception of semantic quality in type: Differences between designers and non-designers," *Information design journal*, vol. 3, no. 1, pp. 38–50, 1982. [Cited on pages 8 and 17.]

[35] J. Nielsen, "Usability inspection methods," in *Conference Companion on Human Factors in Computing Systems*, ser. CHI '94. New York, NY, USA: Association for Computing Machinery, 1994, p. 413–414. [Online]. Available: https://doi.org/10.1145/259963.260531 [Cited on page 8.]

[36] K. Finstad, "The Usability Metric for User Experience," *Interacting with Computers*, vol. 22, no. 5, pp. 323–327, 05 2010. [Online]. Available: https://doi.org/10.1016/j.intcom.2010.04.004 [Cited on page 8.]

[37] J. Brooke, "Sus: A quick and dirty usability scale," *Usability Eval. Ind.*, vol. 189, 11 1995. [Cited on page 8.]

[38] B. Laugwitz, T. Held, and M. Schrepp, "Construction and evaluation of a user experience questionnaire," in *HCI and Usability for Education and Work: 4th Symposium of the Workgroup Human-Computer Interaction and Usability Engineering of the Austrian Computer Society, USAB 2008, Graz, Austria, November 20-21, 2008. Proceedings 4*, vol. 5298. Springer, 11 2008, pp. 63–76. [Cited on pages 8 and 17.]

[39] M. Hassenzahl, M. Burmester, and F. Koller, "Attrakdiff: Ein fragebogen zur messung wahrgenommener hedonischer und pragmatischer qualität," *Mensch & Computer 2003: Interaktion in Bewegung*, pp. 187–196, 2003. [Cited on pages 8 and 17.]

[40] A. B. Kocaballi, L. Laranjo, and E. Coiera, "Measuring user experience in conversational interfaces: A comparison of six questionnaires," in *Proceedings of the 32nd International BCS Human Computer Interaction Conference 32*, 07 2018. [Cited on page 8.]

[41] K. Hone, U. Ph, R. Graham, and A. Link, "Towards a tool for the subjective assessment of speech system interfaces (sassi)," *Natural Language Engineering*, vol. 6, 07 2000. [Cited on page 8.]

[42] I. Díaz-Oreiro, G. López, L. Quesada, and Guerrero, "Standardized questionnaires for user experience evaluation: A systematic literature review," *Proceedings*, vol. 31, p. 14, 11 2019. [Cited on page 8.]

[43] J. Nielsen, "Enhancing the explanatory power of usability heuristics," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '94. New York, NY, USA: Association for Computing Machinery, 1994, p. 152–158. [Online]. Available: https://doi.org/10.1145/191666.191729 [Cited on pages 9 and 17.]

[44] ——, "Why you only need to test with 5 users," Mar 2000. [Online]. Available: https://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/ [Cited on pages 9 and 17.]

[45] R. A. Virzi, "Refining the test phase of usability evaluation: How many subjects is enough?" *Human Factors: The Journal of Human Factors and Ergonomics Society*, vol. 34, pp. 457 – 468, 1992. [Online]. Available: https://api.semanticscholar.org/CorpusID:59748299 [Cited on page 9.]

[46] M. Hertzum and N. Jacobsen, "The evaluator effect during first-time use of the cognitive walkthrough technique." in *HCI (1)*, 01 1999, pp. 1063–1067. [Cited on page 9.]

[47] W. Hwang and G. Salvendy, "Number of people required for usability evaluation: The 10±2 rule," *Commun. ACM*, vol. 53, no. 5, p. 130–133, may 2010. [Online]. Available: https://doi.org/10.1145/1735223.1735255 [Cited on pages 9 and 17.]

[48] R. Queiros and J. Leal, "Orchestration of e-learning services for automatic evaluation of programming exercises," *Journal of Universal Computer Science*, vol. 18, p. 1454textendash1482, 06 2012. [Cited on page 11.]

[49] J. Ramirez-Echeverry, F. Restrepo-Calle, and F. González, "Uncode: Interactive system for learning and automatic evaluation of computer programming skills," in *EDULEARN18 Proceedings*, 07 2018. [Cited on page 11.]

[50] J. Caiza and J. Del Alamo, "Programming assignments automatic grading: Review of tools and implementations," *Inted 2013 Proceedings*, pp. 5691–5700, 01 2013. [Cited on page 11.]

[51] R. Queiros and J. Leal, "Programming exercises evaluation systems - an interoperability survey," in *International Conference on Computer Supported Education*, vol. 1, 01 2012, pp. 83–90. [Cited on page 11.]

[52] M. Piteira, C. Costa, and M. Aparicio, "Computer programming learning: How to apply gamification on online courses?" *Journal of Information Systems Engineering & Management*, vol. 3, 04 2018. [Cited on page 12.]

[53] G. Kiryakova, N. Angelova, and L. Yordanova, "Gamification in education," in *Proceedings of 9th international Balkan education and science conference*, 10 2014. [Cited on page 12.]

[54] R. Queiros, J. Leal, and J. Campos, "Sequencing educational resources with seqins," *Computer Science and Information Systems*, vol. 11, pp. 1479–1497, 10 2014. [Cited on page 12.]

[55] J. P. Leal, R. Queirós, P. Ferreirinha, and J. Swacha, "A Roadmap to Convert Educational Web Applications into LTI Tools," in *Third International Computer Programming Education Conference (ICPEC 2022)*, ser. Open Access Series in Informatics (OASIcs), A. Simões and J. a. C. Silva, Eds., vol. 102. Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022, pp. 12:1–12:12. [Online]. Available: https://drops.dagstuhl.de/opus/volltexte/2022/16616 [Cited on pages 12 and 63.]

[56] E. Baniassad, L. Zamprogno, B. Hall, and R. Holmes, "Stop the (autograder) insanity: Regression penalties to deter autograder overreliance," in *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*, ser. SIGCSE '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 1062–1068. [Online]. Available: https://doi.org/10.1145/3408877.3432430 [Cited on pages 12 and 17.]

[57] T. Lehtinen, A. Lukkarinen, and L. Haaranen, "Students struggle to explain their own program code," in *Proceedings of the 26th ACM Conference on Innovation and Technology in Computer Science Education V. 1*, ser. ITiCSE '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 206–212. [Online]. Available: https://doi.org/10.1145/3430665.3456322 [Cited on pages 12 and 17.]

[58] C. Kennedy and E. T. Kraemer, "Qualitative observations of student reasoning: Coding in the wild," in *Proceedings of the 2019 ACM Conference on Innovation and*

Technology in Computer Science Education, ser. ITiCSE '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 224–230. [Online]. Available: https://doi.org/10.1145/3304221.3319751 [Cited on pages 12 and 17.]

[59] J. Finnie-Ansley, P. Denny, B. A. Becker, A. Luxton-Reilly, and J. Prather, "The robots are coming: Exploring the implications of openai codex on introductory programming," in *Proceedings of the 24th Australasian Computing Education Conference*, ser. ACE '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 10–19. [Online]. Available: https://doi.org/10.1145/3511861.3511863 [Cited on page 13.]

[60] A. L. Santos, "Shifting programming education assessment from exercise outputs toward deeper comprehension," in *4th International Computer Programming Education Conference (ICPEC 2023)*, ser. Open Access Series in Informatics (OASIcs), R. A. Peixoto de Queirós and M. P. Teixeira Pinto, Eds., vol. 112. Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023, pp. 1:1–1:5. [Online]. Available: https://drops.dagstuhl.de/opus/volltexte/2023/18497 [Cited on pages 13 and 17.]

[61] N. Zanjani, S. L. Edwards, S. S. Nykvist, and S. Geva, "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*, vol. 33, pp. 19–31, 2016. [Cited on pages 16 and 17.]

[62] A. Saleh, H. Abu Addous, I. Alansari, and O. Enaizan, "The evaluation of user experience on learning management systems using ueq," *International Journal of Emerging Technologies in Learning (iJET)*, vol. 17, pp. 145–162, 04 2022. [Cited on pages 16 and 17.]

[63] W. Nakamura, E. Teixeira de Oliveira, and T. Conte, "Usability and user experience evaluation of learning management systems - a systematic mapping study," in *Proc. of the 19th International Conference on Enterprise Information Systems*, 01 2017, pp. 97–108. [Cited on page 16.]

[64] R. Queirós, M. Pinto, and T. Terroso, "User Experience Evaluation in a Code Playground," in *Second International Computer Programming Education Conference (ICPEC 2021)*, ser. Open Access Series in Informatics (OASIcs), P. R. Henriques,

F. Portela, R. Queirós, and A. Simões, Eds., vol. 91. Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021, pp. 17:1–17:9. [Online]. Available: https://drops.dagstuhl.de/opus/volltexte/2021/14233 [Cited on page 20.]

[65] A. Issac, "Chatgpt functions malformed json," Jul 2023. [Online]. Available: https://community.openai.com/t/chatgpt-functions-malformed-json/306509 [Cited on page 48.]

[66] E. D. Experts, "How to use chat gpt: A simple guide for beginners," Sep 2023. [Online]. Available: https://blog.enterprisedna.co/how-to-use-chat-gpt/ [Cited on page 48.]

[67] L. Ruettimann, "How to interact with chat gpt-4 effectively: Top tips for better questions," Sep 2023. [Online]. Available: https://laurieruettimann.com/chat-gpt-4-ask-questions/ [Cited on page 48.]

[68] J. Martinez, "Return json from gpt," Jul 2023. [Online]. Available: https://betterprogramming.pub/return-json-from-gpt-65d40bfc2ef6 [Cited on page 49.]