UNIVERSITY OF TARTU
Institute of Computer Science
Software Engineering

Anneli Klamas

Quiz Converter: The tool for creating quizzes in Coursera and Moodle

Master's Thesis (30 ECTS)

Supervisor(s): Jaak Vilo, PhD

Tartu 2024

Quiz Converter: The tool for creating quizzes in Coursera and Moodle

Abstract:

This thesis describes the development of an application that can convert DOCX file that follows a specific format into Moodle XML and Coursera DOCX. As few open-source tools can accomplish this, the application developed throughout this thesis will help teachers minimise manual work while creating the quiz questions.

The Quiz Converter application was created by two developers where, one focused mainly on the backend and deployment part of the application and the other one on the frontend.

Keywords:

Java application, document converter, Moodle, Coursera, Java, Spring, Apache POI, Docker, DOCX, XML

CERCS: P170 Computer science, numerical analysis, systems, control

Quiz Converter: tööriist testide loomiseks Courseras ja Moodle'is

Lühikokkuvõte:

Käesolevas lõputöös kirjeldatakse konkreetset vormingut järgivat DOCX-faili Moodle XML-iks ja Coursera DOCX-iks teisendava rakenduse arendamist. Kuna selle eesmärgi täitmiseks on vähe tasuta tööriistu, aitab selle lõputöö käigus arendatud rakendus õppejõududel testide loomisel käsitsi tööd vähendada.

Rakenduse Quiz Converter lõid kaks arendajat, kellest üks keskendus peamiselt rakenduse tagasüsteemile ja paigaldamisele ning teine selle eesliidesele.

Võtmesõnad:

Java rakendus, dokumentide teisendaja, Moodle, Coursera, Java, Spring, Apache POI, Docker, DOCX, XML

CERCS: P170 Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine (automaatjuhtimisteooria)

Table of Contents

Introduction

Schools use different platforms to share materials and check students' progress interactively through tests. Both TalTech and the University of Tartu use Moodle and Coursera to achieve this purpose. While both platforms provide a suite of different tools and capabilities, they lack the possibility for course administrators to work on the same test concurrently. This means that it is not that easy to spot mistakes and suggest improvements to each other. Therefore, teachers use Google Drive or Microsoft Cloud for collaboration to create .docx files, which support comments, working together simultaneously on the same file and history of changes.

Problems also arise when the course administrator decides to switch from one platform to another. Both previously mentioned services allow for exporting and importing tests, but they use their own file formats with little overlap. This means that when a platform switch is decided upon, a large amount of work is required to learn the new platform and recreate all the courses and tests.

The main problem is understanding how to create an application that would read in the .docx file with the questions written in a specific format and then output a file that Coursera or Moodle recognise. From that, the following questions are asked:

- What restrictions are there on Coursera and Moodle?
- How to define and display errors in an uploaded file to the user?
- What similar applications are there?

This thesis aims to build a web application for converting simple collaboratively edited Word document files from platforms like Google Docs to the file formats that Coursera and Moodle both accept. To accomplish this task, the author will also research and implement the necessary technology stack for the backend, including testing and deployment suites required for software development.

The thesis is divided into seven chapters. The first chapter overviews the Coursera and Moodle learning management systems. The second chapter introduces other applications that that help with importing questions to the Moodle. The planning process of the Quiz Converter is described in the third chapter. The fourth chapter focuses on the what was created during the

development process. The fifth chapter describes how the application was tested. The sixth chapter provides the summary of the done work and introduces the future work.

1. Learning management system

This chapter will introduce two LMSs: Moodle and Coursera. A learning management system, in short, an LMS, is a system that helps teachers keep track of the courses. With LMS, teachers can share their documents, create quizzes that can be automatically and or manually graded, keep track of the students' grades and progress, share feedback, and communicate with students.

1.1. Moodle

According to the Moodle's webpage about their history [1] Moodle was created by Martin Dougiamas. The name came from the acronym of Martin's Object-Oriented Dynamic Learning Environment, which was later changed to Modular Object-Oriented Dynamic Learning Environment. In 2001, Moodle came to live with the first Moodle course, "Constructivism" which run at Curtin University. Within a few months, Moodle was being used all over the world. Now, 23 years later, Moodle has over 377,000,000 users, 2.2 billion course enrolments, 45,000,000 courses in 42 languages, and 154,000 Moodle sites [2].

1.1.1. Question bank

Moodle's official documentation [3] states that question bank allows teachers to store, create, preview, and edit questions, which can then be used in the quizzes. Questions can be organised into different categories and subcategories for better management. Questions can be added to the question bank either by creating new ones through Moodle's user interface or by importing them from different file formats (Table 5).

However creating questions in collaboration with somebody is hard, as there is no option to comment on the questions or see history of changes. Therefore, teacher prepare their questions in advance using Microsoft Word files and then create each question manually in the question bank.

1.1.2. Creating quizzes in Moodle

According to the Moodle's documentation page about quizzes [4] there are three ways to add questions to the quiz. One way is to create a new question, which will not be added to the question bank after the creation. The second way is to import question(s) from the question

bank. The third option is to add random questions, which means that question tags and the number of questions to be imported can be specified. So, if students are taking the same quiz, then they will have different questions on their quiz.

1.2. Coursera

Coursera was founded in 2012 by Daphne Koller and Andrew Ng. It has over 113 million learners, 7000 campuses, businesses, and governments [5].

1.2.1. Quiz item vs assignment items

Coursera's official documentation page [6] states that in February of 2023, Coursera launched assignment items. Assignments items combine features of Coursera quiz items and legacy staff graded assignments into one. Compared to quiz items new assignment item has same auto-graded question types as quiz items as well as multiple dropdowns question types. The manually graded questions and instructions for learners and grades can be added to the assignments but not to the quiz items. Question banks are not supported by the quiz items and can only be used with assignments. Other features that are supported by the assignments but not by quiz items are: choice of latest test score to be counted towards the grade, time limits to public tests, auto-graded assignments can have submission limit per timed attempts, student's grade can be visible or hidden and the submission can be hidden from the learner.

However the Coursera's article about assignment items [6] states that not all features thatare supported by quizzes are supported by assignment item. The new assignment items can not have extra credit questions and assignment items are not included in course assessment dashboard.

In February 2024, the Coursera support page stated [7] that the creation of new quiz items is no longer supported and, by the end of 2024, will be fully removed. The assignment items should be used to create quizzes instead of quiz items.

1.2.2. Question bank

Question banks were launched so that teachers could tag questions based on the difficulty and objectives and reuse the questions in different assignments [7]. However questions from the question banks can not be used in the Coursera quiz items as they are only supported by assignments [8].

In Coursera, according to its documentation page [9], there are three ways to create questions for the question banks. The first option is to import it from a specified file (Table 6). The second option is to create them through their user interface. The third option, however, is to create a question with AI (Artificial Intelligence) automatically. Currently this feature is beta release and is available for all the educators. It uses OpenAI technology to automatically generate questions based on learning objectives, lesson titles, videos and text material included in the course modules and lessons.

1.3. Supported question types in Moodle and Coursera

Table 1 shows the comparison of the supported question types in Moodle and Coursera. If the question type is not supported but the same result can be achieved by using another question type, then in the table, it is marked as supported. Moodle has more question options than Coursera. Both have different plugins that can be added to support more question types. The main difference is that Moodle supports drag-and-drop questions, and Coursera supports more dropdown question types. Coursera also supports code expression type questions where students can write code, and it can be automatically evaluated. However, the teacher needs to write scripts to assess and grade the output of the students' code. Coursera has only auto-graded question types in its quizzes.

Table 1. Coursera and Moodle question type comparison.

	Meaning	**Moodle**	**Coursera**
Calculated	Numbers can be defined as variables and will be autogenerated with every quiz. The answer is defined as a formula with variables. Students still need to provide numerical answers.	Yes	No
Calculated multi-choice	Multi-choice question with autogenerated variable values and answers are defined as formulas. Students can choose the correct numeric answers.	Yes	No

Calculated simple	Calculated question with a more straightforward creation interface.	Yes	No
Drag and drop into the text	Dragging missing words or phrases into text.	Yes	No
Drag and drop markers	Drag and drop answer options to the image without visible drop zones.	Yes	No
Drag and drop onto image	Drag and drop answer options to the image with visible drop zones.	Yes	No
Description	It is not actually a question but a text and/or graphic that can be used to introduce the next question group.	Yes	No
Essay	The manually graded question allows students to write long text answers, such as essays.	Yes	Yes
Matching	The question consists of two lists that must be matched against each other. For example, “Match the author to the book title”.	Yes	Yes* only in Coursera assignments
Embedded Answers (Cloze Test / Gap Fill)	The question which can consist of multiple different questions with different question types.	Yes	No
Multiple choice	Question type where students can choose answers from the multiple answer options. It	Yes	Yes

	can be either a single correct answer (radio button) or multiple correct answers (checkbox).		
Ordering	Displays several items, whether images, phrases, or images, in a random order. Students can drag them to put them in the proper order.	Yes	No
Short Answer	The answer must match a word or a phrase.	Yes	Yes
Numerical	The answer can be a numerical value. The accepted error can be set.	Yes	Yes
Random short-answer matching	Looks just like a matching question, but the sub-questions are randomly drawn from the short answer questions in the current category [10].	Yes	No
Select missing words	Select the missing word from the dropdown in a text.	Yes	Yes* only in Coursera assignments
True/False	Two options with a single correct answer	Yes	Yes
Multiple dropdowns categorisation	Question has category columns that can be filled using dropdowns. For example there can be two columns: plants and animals. Dropdowns then have options: mouse, cat, dandelion and rose. The correct options must be selected under the correct category.	No	Yes* only in Coursera assignments

Math expression	Question that can be answered by entering mathematical formulas and/or constants.	No	Yes
Regular expression	The question where teacher can define the answer by using regular expression. Any word or phrase that will match the regular expression will be count as correct.	No	Yes
Code expressions	There is a code editor where students can write and run the code before submitting it.	No	Yes
Reflective multiple choice	Multiple-choice question where all the answers are correct.	Yes	Yes
Reflective single choice	Single choice question where all the answers are correct.	Yes	Yes
Reflective text answer	Essay-type question where all the answers are automatically graded as correct.	No	Yes

2. Competitor Analysis

As it is possible to import .docx files directly to Coursera, then it is easy for the teachers to follow their formatting rules. This cannot be said about Moodle, which does not support Microsoft Word or Microsoft Excel files. That is why people have come up with different products that would help them. For these reasons, this chapter will focus only on products that deal with Moodle import options.

Table 2 shows which products there are that help with importing Moodle files. The first column of the table shows the name of the product. The second column, "Type", describes what type of product it is, whether it is a service, web application, Moodle plugin, or Microsoft Excel workbook. The table also contains information about the price, as some of the products are not open source. The "Import file format" and "Export file format" columns show which file formats the product takes in and which are its output formats that can be imported into Moodle. There is also a description of the purpose of the products, whether it is to convert text files to GIFT, migrate the whole course to another learning management system, or some other purpose.

Table 2. Products for importing Moodle quizzes.

Product	**Type**	**Price**	**Import file format or extension**	**Export file format or extension**	**Purpose**
Scaffold Migration by K16 Solutions	Automated service	Based on the price request	N/A	N/A	Migrating the whole course from one LMS to another

GetMarked	Web application	From 29 USD per year	16 different file formats, including .docx and .pdf	27 different platforms, including Moodle, Coursera, and Kahoot!	Can upload files with quizzes, and then it uploads to the chosen platform
Moodle XML converter	Web application	Open source	.txt	Moodle XML	Converts .txt quiz files to Moodle XML file
FastTest Plugin	Moodle plugin	Open source	.xlsx	Moodle XML	Template based on Microsoft Excel for creating Moodle quizzes
moodle2word	Moodle plugin	Open source	.docx	.docx	Allows import questions from .docx file to Moodle
Moodle test creator	Web application	Open source	Text	GIFT	Converts text to GIFT to import into Moodle
Moodle Cloze and GIFT	Microsoft Excel workbook	Open source	Text	GIFT	Excel workbook, which converts its

Code Generator					content to the GIFT format

Table 3 shows what question types the products that were mentioned in the Table 1 support and whether they support images. As Scaffold Migration by K16 Solutions claims to be able to move all the course information with its questions from the question banks, but different platforms support slightly different question types then, without using the product, it couldn't be determined if all the question types are supported and what would happen if the platform A with question type B would be moved to the platform C which wouldn't support the question type B. For that reason, all the question types are marked as N/A.

Table 3. Product comparison by support question type.

	Scaffold Migration by K16 Solutions	**GETMARKED**	**Moodle XML converter**	**FastTest Plugin**	**moodle2word**	**Moodle test creator**	**Moodle Cloze and GIFT Code Generator**
Images	N/A	N/A	No	Yes	Yes	No	No
True-false	N/A	Yes	Yes	Yes	Yes	Yes	Yes
Multiple choice	N/A	Yes	Yes	Yes	Yes	Yes	Yes
Multiple response	N/A	Yes	Yes* (doesn't allow having only one correct	Yes	Yes	Yes	Yes

			answer)				
Cloze	N/A	N/A	Yes	Yes	Yes	No	Yes
Open ended/essay	N/A	Yes	Yes	Yes	Yes	Yes	Yes
Matching	N/A	Yes	Yes	Yes	Yes	Yes	Yes
Regular Expression	N/A	No	No	No	Yes	No	No
Calculated	N/A	No	No	No	No	No	No
Numerical	N/A	No	No	No	No	Yes	Yes

2.1. Scaffold Migration by K16 Solutions

Scaffold Migration is according to their official website [11] an automated service provided by K16 Solutions for migrating from one LMS platform to another. If a teacher is moving from Moodle to Coursera, then they can send a price inquiry, and then all their quizzes, questions from the question banks, materials, and course structure will be moved to Coursera without the need for copy-pasting or manually inserting anything.

2.2. GETMARKED

GETMARKED, according to their website [12], [13] is a web application that lets teachers upload their quizzes to their website and then export them to one or multiple of 27 supported platforms. Due to their use of AI to understand the content of the file with the quizzes, there are no exact rules that need to be matched (Figure 1). GETMARKED supports eight different types of questions: True-false question, multiple choice question, multiple response question, cloze question (fill-in-the-blank), cloze question (select dropdown menu), open-ended

question, matching question, and Likert question. Alongside 16 different import file formats, teachers can also import the PDF versions of their tests.

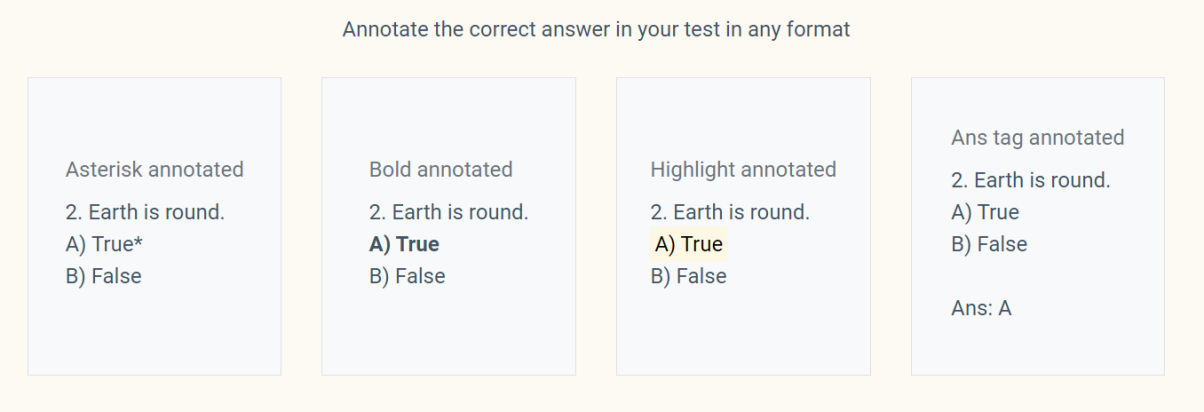


Figure 1. Example of GETMARKED correct answer annotations [12].

2.3. Moodle XML converter

Moodle XML converter, formerly known as VLT tool, is an open-source web application that, according to their documentation [14], based on the rules of the text file format, converts quizzes to Moodle XML file, which can be imported into Moodle. It supports eight different question types: multiple choice, which is converted into radio button type question if there is only one correct answer and checkbox question if there are multiple correct answers; short answer; essay; description; cloze, true-false; numerical; matching; and order, which requires additional Moodle module to be installed. Moodle XML converter does not support images. Nevertheless, it does give the user warnings and error messages if there have been mistakes in the provided text file.

2.4. FastTest Plugin

According to the article written by Milagros Huerta and Manuel Alejandro [15] FastTest Plugin is a Moodle plugin created in January of 2021 by Milagros Huerta from the University of Cádiz and Manuel Alejandro from the University of Cádiz. The plugin is open source and based on a Microsoft Excel spreadsheet. It can be used to create questions in graphical UI (Figure 2) and export them to Moodle. It supports images as well as eight question types: MCQ, True / False, Matching, Missing Word, Essay, Short Answer, Cloze, and Calculated.

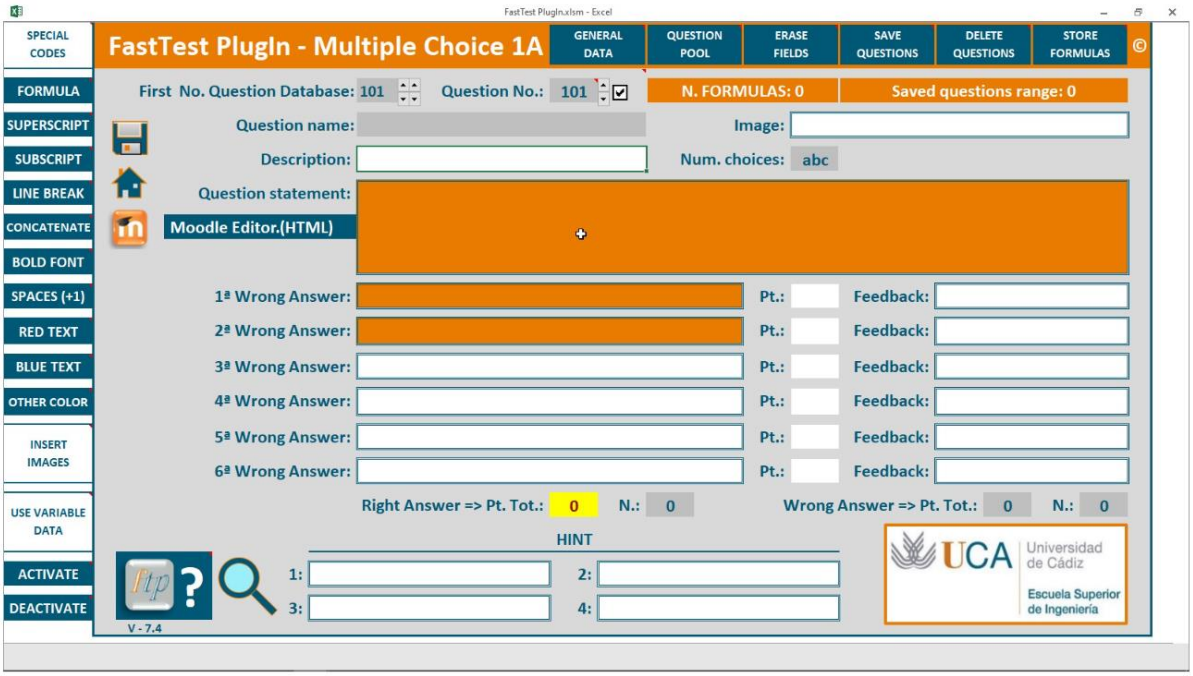


Figure 2. FastTest plugin UI [16].

2.5. Moodle2word

Moodle2word is according to Moodle plugins website [17] an open-source Moodle plugin that helps import Microsoft Word files in .docx format directly to the Moodle course question bank. It also exports Moodle questions to the .docx format so that they can be easily changed in Word and then imported again. The Word file should be in the table format (Figure 3). All the main question types are fully supported except for numerical and calculated question types, which can be exported but cannot be imported. It is also possible to install a Microsoft Word plugin, which will create the table templates itself without the need to format and make them manually [18].

Boolean question - soccer (TF)

Real Madrid beat Liverpool in the Champions League Final 2018. True or False?			*TF*
		Default mark:	1
	Answers	**Feedback**	**Grade**
	True		*100*
	False		*0*
	General feedback:		
	Tags:		
Set grade '100' to the correct answer.			

Figure 3. Boolean question example for Moodle2word [18].

2.6. Moodle Test Creator

As stated on the Moodle Test Creator website [19], Moodle Test Creator is an open-source web application created by Manuel Vilas. The purpose of the app is to convert the imported text into GIFT format, which will be recognised by Moodle. It also gives error messages if the inserted text is in the wrong format or has other mistakes in it. It supports six different question types: single answer, multiple answers, short answer, numerical, essay, and true-false questions. However, it does not support images.

2.7. Moodle Cloze and GIFT Code Generator

Moodle Cloze and GIFT Code Generator based on its website [20] is a Microsoft Excel workbook created by Jordan Svien. The workbook lets users fill in the cell with the question details and then copy the GIFT code to import it into the Moodle question bank (Figure 4).

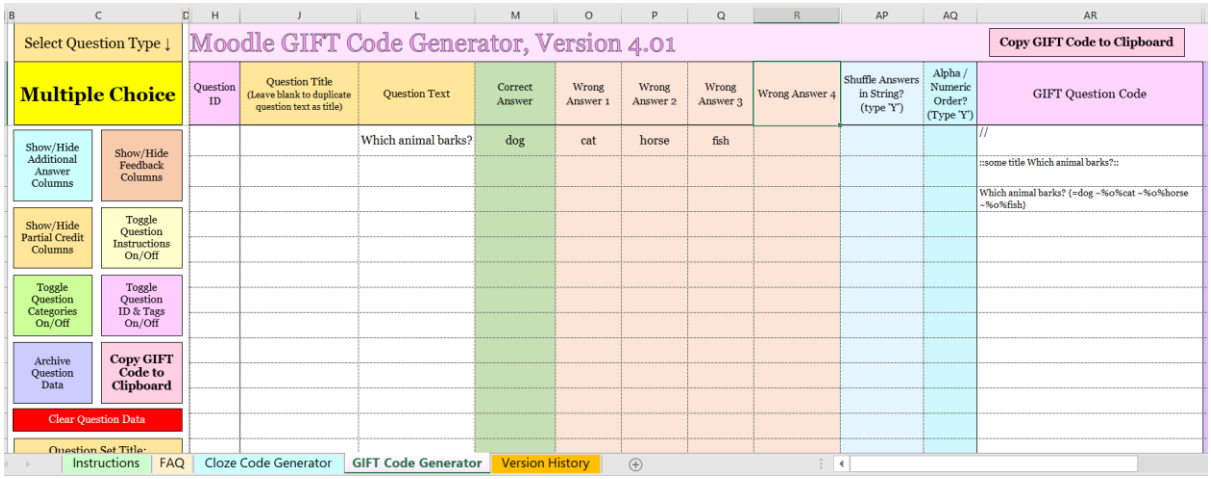


Figure 4. Moodle Cloze and GIFT Code Generator with example question.

2.8. Conclusion

As working on the Moodle quizzes can be challenging, there are many products to ease that pain. However, the ones that are really easy to use and require no changes to the already made quizzes, like GETMARKED and Scaffold Migration by K16 Solutions, are not open source, and you need to have an annual subscription or custom pricing to use them. Open source products can be divided into three groups: based on Microsoft Excel spreadsheets, which require users to use their templates, which are not always intuitive and therefore need time to learn how to use them; text based web applications, which do not support adding images to the

questions; and Microsoft Word based Moodle plugins which require users to install their plugin and follow their Microsoft Word templates. So, there is a need for an application that would not need to install anything, be open source, and have intuitive templates that would not follow a strict format.

3. Building Quiz Converter

The web application was created in cooperation with the author and Bachelor's student Tobias Reiter. The author's focus was mainly on the backend and deployment process, and the Bachelor's student was on the frontend, template, and documentation. The development followed agile development methodology, which will be described in this chapter.

3.1. Collecting requirements

Interviews were conducted to get a better understanding of how teachers create quizzes. The author and codeveloper went to the University of Tartu Delta building and asked six teachers about how they create the quizzes, do they discuss them with other people, how they discuss them, and what they like or dislike about their approaches.

3.1.1. Interview insights

Based on the interviews, the author and codeveloper got these insights:

- Not all people use Moodle. One teacher uses Coursera because they find its user interface to be better than Moodle's. The other one uses Kahoot! for quick and fun knowledge checks for students. One teacher creates self-assessment quizzes in Courses as they have all the course content there.
- Usually, teachers use multiple choice and single choice question types.
- All teachers who were interviewed said that they do not change the default settings of the quizzes in Moodle.
- For some, the pictures are essential, especially when there is no option to format the text.
- One teacher mentioned that it would be nice if the format of the text would also be imported into Moodle. For example, if the .docx contains red text in the question description, then Moodle would also have red text.

3.1.2. Defining requirements

Based on the interview and competitor analysis insights, the author and codeveloper decided to define requirements for MVP (minimum viable product).

- The imported document should be a .docx file, as it is one of the most commonly used among interviewed teachers.
- Single choice (radio button) and multiple choice (checkbox) questions should be supported, as they are the most popular question types
- There should be error messages if there are errors in the document formatting so that teachers can spot their mistakes before trying to import the output document to the chosen LMS.
- Pictures should be supported in question descriptions and answers, as it could be tricky to format code in the text document.
- Converting to Moodle and Coursera should be supported, as they are one of the most popular LMS platforms that the University of Tartu supports.
- The rules for document formatting should be defined and made available for the teachers to see so that they know how to format their documents.

Based on the defined requirements for the MVP, the initial UI designs were created (Figure 5). The University of Tartu's official colour was used as a primary colour. To create the designs, Figma was used as it has easy to use user interface and it is possible to work together on the designs at the same time.

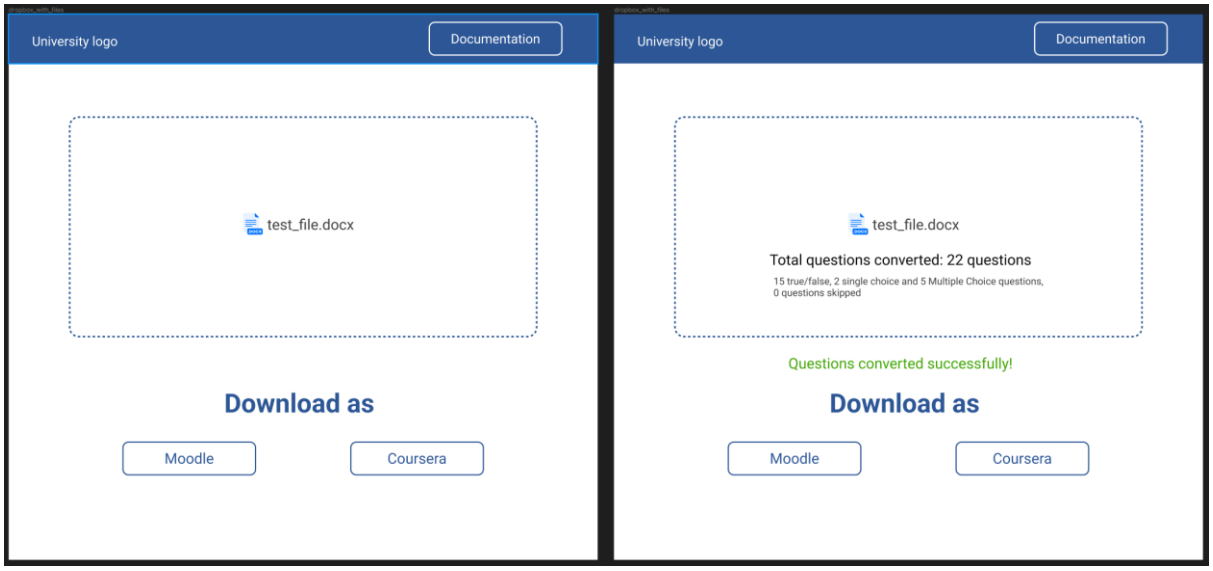


Figure 5. The initial design of the landing page with file before uploading and initial design of the feedback of uploaded file.

3.2. Choosing technologies

After collecting initial requirements for the MVP, the author and codeveloper started discussions about the tech stack to use.

For the backend, the Java language was chosen as it is one of the most popular languages used and it also has multiple libraries for reading in .docx files [21]. In addition, both the author and codeveloper have previous experience with it. Based on the author's prior experiences, the Spring framework was chosen for the backend development.

For the frontend Next.js, the React framework for web development was chosen. Next.js is a framework based on React. It was chosen as it has more functionalities than plain React. Some of the key features of the Next.js according to the GeeksforGeeks tutorial [22] are:

- server-side rendering,
- static site generation,
- automatic code splitting,
- data fetching,
- routing,
- image optimisation,
- built-in CSS and JavaScript bundling,
- API routes.

These built-in features make code shorter and more manageable, and the web pages load faster.

3.2.1. Backend libraries for DOCX

Java has multiple libraries for parsing .docx files. The main ones are Aspose, Docx4J, and Apache POI (Table 4).

Table 4. Comparison of different libraries for parsing .docx files.

Library	Pricing	Documentation	First release year	Last release
Aspose	From 1199$ (one-time payment)	Yes	N/A	N/A

	[23]			
Docx4J	Open source	Yes	N/A	25th of April 2024
Apache POI	Open source	Yes	28th of August 2001	25th of November 2023

As Aspose is not open source, it was not even taken into consideration when choosing the library. Apache POI was selected as it is maintained, vulnerability issues are being addressed and fixed, and there is a lot of support from the community, so information and tutorials are easy to find, and it has documentation. The other option was to use Docx4J as it is especially designed for .docx processing compared to Apache POI, which primary purpose is working with Microsoft Excel files. However, Docx4J is not as transparent with their vulnerability issues and their fixes.

3.2.2. Apache POI

Apache POI is part of the Apache Software Foundation. POI's first release was on the 28th of August 2001 [24]. Apache POI is used to create, edit, and read Excel, PowerPoint, Word, Publisher, OLE2 Filesystem, and OLE2 Document Props files in Java [25].

3.2.2.1. Apache POI for Word

Based on Apche POI's website [26] Apache POI has two versions for processing Word files. The HWPF is a version for Microsoft Word 97(-2007) file format. For the new Word 2007 .docx format, the XWPF is used. Both HWPF and XWPF have similar features. Apache describes them as "moderately functional" as the text extraction support is very strong, but the editing and creation of the Word files may be limited or incomplete.

One of the advantages of Apache POI is that it has its own exeptions if the file is not .docx or if the file is too long, so that it is not possible to read files that may contain malicious content or to use too much of the machine's resources if somebody uploads a file that is big. Another advantage is that with Apache POI it is easy to get the paragraphs from the .docx file. Figure 6 shows how to get paragraphs from the MultipartFile. Firstly the new XWPFDocument needs to be created from the file's input stream of bytes, then getParagraphs() should be called, which returns list containing XWPFParagraphs.

```java
public List<Question> convertDocToQuestion(MultipartFile file) throws IOException {
    XWPFDocument docx = new XWPFDocument(file.getInputStream());

    var state = new QuestionState();
    var questions = new ArrayList<Question>();

    docx.getParagraphs().forEach(paragraph -> handleParagraph(state, questions, paragraph));

    var validationHandler = new QuestionValidationHandler(state);
    validationHandler.validateQuestion();
    questions.add(state.createQuestion());

    return questions;
}
```

Figure 6. Getting paragraphs from the MultipartFile using Apache POI.

Figure 7 shows that getting images from the paragraphs is not as easy as getting text with getParagraphText() as there is no method like getParagraphImages(). Images can be retrieved through the XWPFRun object, which defines a text region with a common set of properties [27]. Once the XWPFRun is returned, it is possible to use run.getEmbeddedPictures(), which will give a list of pictures from the paragraph run. With the XWPFPicture object, it is possible to encode the data of the picture to the base64 to use it in the HTML when creating Moodle XML or convert it back to the XWPFPicture when creating a new .docx file (Figure 8). XWPFPicture also contains information about the width and height of the image, so it can be added to the .xml with the original dimensions.

```java
private List<Picture> handleParagraphPictures(XWPFParagraph paragraph) {
    var runsWithPictures : List<XWPFRun> = paragraph.getRuns().stream()
            .filter(r -> !r.getEmbeddedPictures().isEmpty()).toList();
    var paragraphPictures = new ArrayList<Picture>();
    if (!runsWithPictures.isEmpty()) {
        for (var run : runsWithPictures) {
            var pictures : List<XWPFPicture> = run.getEmbeddedPictures();
            for (var picture : pictures) {
                var data : String = Base64.getEncoder()
                        .encodeToString(picture.getPictureData().getData());
                var name : String = picture.getPictureData().getFileName()
                        .replace(picture.getPictureData().getPictureTypeEnum().extension, replacement: "");
                paragraphPictures.add(new Picture(
                        data, picture.getWidth(), picture.getDepth(), name, picture.getPictureData().getPictureTypeEnum()
                ));
            }
        }
    }
    return paragraphPictures;
}
```

Figure 7. Getting paragraph images using Apache POI.

```
private static void addPictures(Picture p, XWPFDocument doc) {
    var pictureParagraph :XWPFParagraph = doc.createParagraph();
    var pictureRun :XWPFRun = pictureParagraph.createRun();
    var pictureData :byte[] = Base64.getDecoder().decode(p.base64());
    try {
        pictureRun.addPicture(
                new ByteArrayInputStream(pictureData),
                p.type(),
                p.name(),
                Units.toEMU(p.width()),
                Units.toEMU(p.height()));
    } catch (InvalidFormatException | IOException e) {
        throw new RuntimeException(e);
    }
}
```

Figure 8. Creating XWPFPicture and adding it to the paragraph using Apache POI.

With apache POI it is also possible to get the dropdowns and their selected values from .docx files (Figure 9).

```
private List<String> handleDropdownLists(XWPFParagraph paragraph) {
    return paragraph.getIRuns().stream()
            .filter(r -> r instanceof XWPFSDT)
            .map(r -> ((XWPFSDT) r).getContent().getText())
            .toList();
}
```

Figure 9. Getting dropdown values from paragraph using Apache POI.

3.3. Moodle import files

Moodle supports multiple import file formats that are introduced in Table 5 [28].

The Moodle XML format was chosen as the output for Moodle as it is very comprehensive and supports images out of the box without the need to add plugins. In addition, it has excellent documentation. Moodle XML is the default export type for the questions, so it is possible to create a question in Moodle and download it as Moodle XML to see how it is defined. Therefore, it is convenient for development and debugging purposes as developers can compare their Moodle XML to the one that is generated by Moodle.

Table 5. Supported import file types in Moodle.

	Supports images	**Example**
AIKEN	No	.txt file `What is the correct answer to this question?` `A. Is it this one?` `B. Maybe this answer?` `C. Possibly this one?` `D. Must be this one!` `ANSWER: D` *Figure 10. Example of AIKEN file [29].*
Blackboard	Yes	.dot file
GIFT	Yes* (plugin needed)	.txt file `// multiple choice with specified feedback for right and wrong answers` `::Q2:: What's between orange and green in the spectrum?` `{ =yellow # right; good! ~red # wrong, it's yellow ~blue # wrong, it's yellow }` *Figure 11. GIFT format file content example [30].*
Embedded Answers (Cloze)	N/A	.txt file `Which animal eats mice? (answer: the cat)` `Testing MULTICHOICE {1:MULTICHOICE:=the cat#Yes, that's right~the dog#Nope!}` `Testing MULTICHOICE_H {1:MULTICHOICE_H:=the cat#Yes, that's right~the dog#Nope!}` `Testing MULTICHOICE_V{1:MULTICHOICE:=the cat#Yes, that's right~the dog#Nope!}` *Figure 12. Cloze format example [31].*
Moodle XML	Yes	.xml file

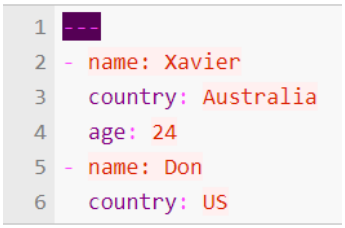
		<answer fraction="100"> <text>The correct answer</text> <feedback><text>Correct!</text></feedback> </answer> <answer fraction="0"> <text>A distractor</text> <feedback><text>Ooops!</text></feedback> </answer> <answer fraction="0"> <text>Another distractor</text> <feedback><text>Ooops!</text></feedback> </answer> <shuffleanswers>1</shuffleanswers> <single>true</single> <answernumbering>abc</answernumbering> *Figure 13. Moodle XML example [32].*
Missing word	No	.txt file As soon as we begin to explore our body parts as infants we become students of {=anatomy and physiology ~reflexology ~science ~experiment}, and in a sense we remain students for life. *Figure 14. Missing word file example [33].*

3.4. Coursera files

Coursera has multiple different import file types for importing quiz items [34], [35]. The import file types with their examples and whether they support images are provided in Table 6. DOCX was chosen as an import file type as it is Coursera's recommended file type and it is the only file format that accepts images. As well as the only format that assignment items accept [36].

Table 6. Supported import file types in Coursera.

	Supports images	**Example**
DOCX	Yes	.docx file

		Multiple Correct Answers **Question 1 - checkbox, shuffle, partial credit** Question prompt goes here A: Incorrect answer Feedback: Add feedback about why this option is incorrect *B: Correct answer Feedback: Add feedback about why this option is correct *C: Correct answer 2 Feedback: Add feedback about why this option is correct D: Incorrect answer 2 Feedback: Add feedback about why this option is incorrect *Figure 15. Example of DOCX file format [37].*
YAML	No	.yaml file  1 --- 2 - name: Xavier 3 country: Australia 4 age: 24 5 - name: Don 6 country: US *Figure 16. Example of YAML file [38].*
QTI	No	.txt, rtf, .doc, .docx, .csv, and StudyMate Class format (.zip and .xml) **Type: MA** **3) Which of the following individuals are credited with determining the exact speed of light?** **a. Albert Einstein** ***b. Albert Michelson** **c. Thomas Edison** ***d. Edward Williams Morley** *Figure 17. Example of QTI file format [39].*

3.5. Development process

The author and co-developer used agile methodology in their development process, as described in Figure 18.

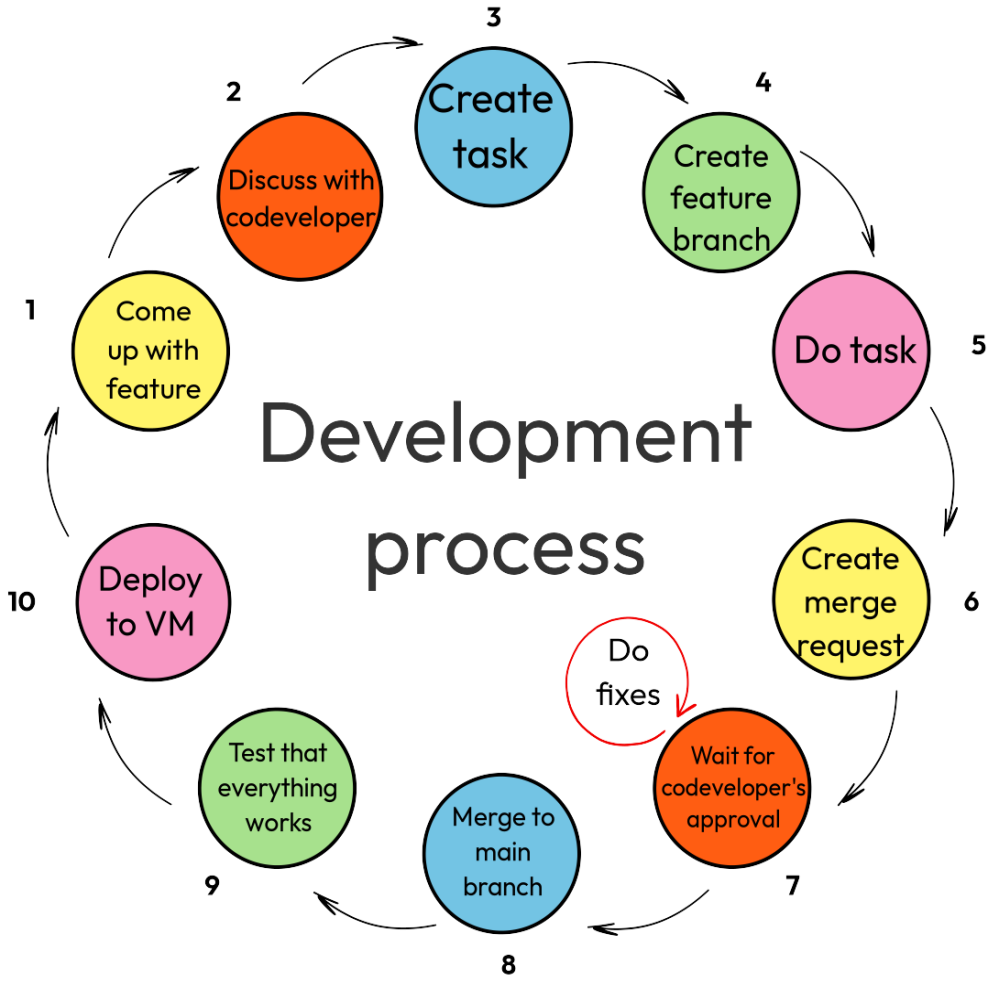


Figure 18. Development process.

During the development process, the author and co-developer updated the feature list when they came up with new ideas for improvements. They discussed whether the feature should be prioritised or not and added according to tasks. Then, when the developer was ready to take on a new task, they took the one with the top priority, created a new branch, and started working on it. Then, they implement the task on their feature branch. Once it is implemented and tested, the merge request is created to merge the feature branch to the main branch. After that, the other developer is notified. The other developer then takes a look at the changes made and, if needed, suggests some improvements. When there are no more suggestions from the developer

who reviewed the merge request, then the merge request is merged. Then, the main branch is tested, and if everything works as expected, then the code is deployed to the virtual machine.

4. Quiz Converter

This chapter describes the end result of the practical part of this thesis. It gives an overview of the overall architecture of the Quiz Converter as well as how all the components work in the backend. It also gives an overview of the flow in the frontend and the list of the implemented features.

4.1. Architecture

The Quiz Converter's architecture is based on the MVC (Model-View-Controller) pattern. MVC pattern has different implementations where in the initial and most common one model can interact directly with the view (Figure 19) [40].

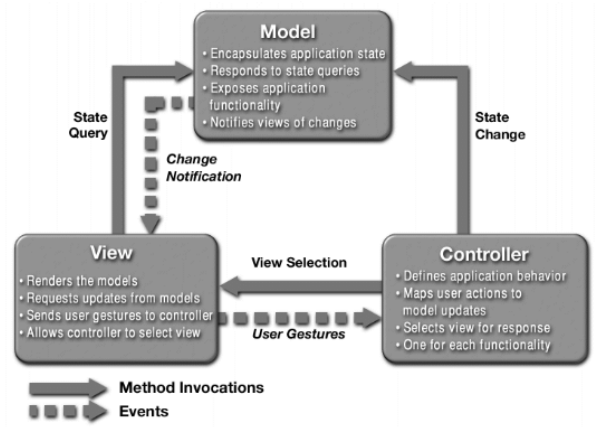


Figure 19. A common MVC Implementation [40].

Oracle's webpage about MVC architecture [40] states that in more recent implementation, the controller is placed between the model and view so that the communication between the view and model is always through the controller (Figure 20). One thing in common is that every implementation divides an application into three main categories: Models, Views, and Controllers, and each of the categories has its own responsibility.

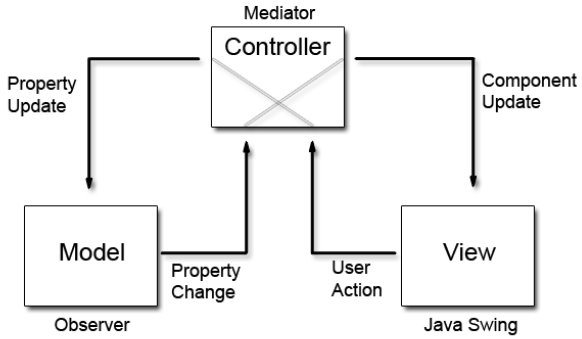


Figure 20. MVC pattern with controller between view and model. [40]

According to the Microsoft's page [22] The model's responsibility is to handle all the business logic, manipulate data, and represent the state of the application. The view's responsibility is to present data from the model to the user; therefore, it should not contain any business logic inside of it. The controller's purpose is to react to the user's interactions, which, for example, can be button clicks in a stand-alone GUI client or HTTP requests in an enterprise web application [21].

4.2. Quiz Converter MVC implementation

Figure 21 shows the Quiz Converter architecture, which follows the MVC pattern that places the controller in the middle of the model and view. The application is divided into two parts: the frontend part and the backend part. The frontend part is the view. The backend part contains the controller and model.

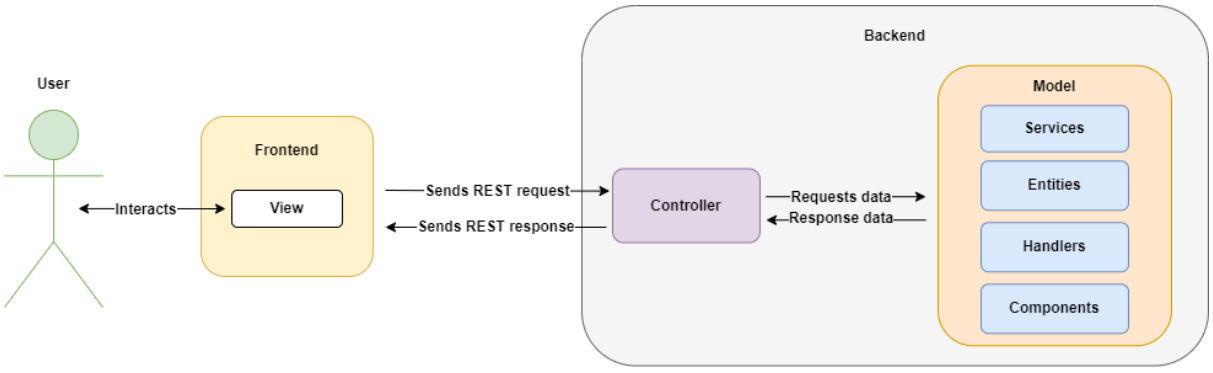


Figure 21. MVC pattern as applied in Quiz Converter.

User can interact with the view by uploading the .docx file and then clicking on the button to convert it to the Coursera formatted .docx or to the Moodle XML file. Once the button is

clicked, the HTTP POST request with the body that contains a file is sent to the backend, where the controller recognises it and forwards the file to the model layer. In the model layer, the file is read as a question object and validated. The details of the file content are added, and the question object is converted into Moodle XML or Coursera .docx file. After that, the file with its details is sent back to the controller where the file DTO (data-transfer-object) is created and sent back to the view as REST ok response with file DTO in the body. The view automatically downloads it and shows question details in the pop-up modal.

4.3. How the backend works

Frontend sends POST /file/convert/moodleXML or POST /file/convert/courseraDocx request to the backend, where the controller recognises it, takes out the .docx file from the request body, and then sends the MultipartFile to the ConverterService. ConverterService has two methods: convertDocxToMoodle and convertDocxToCoursera, both of which take a file as an input argument (Figure 22). Depending on which POST request was sent, the controller calls one of those methods. Then CoverterService calls FileUploadService, which is responsible for reading the .docx file paragraph by paragraph. Based on how each paragraph starts or which was the previous paragraph, it is assigned a type: question details, question description, answer option, answer option feedback, empty text, default feedback, or unknown. Based on the paragraph type, the paragraph is skipped, QuestionState is updated, and/or the new Question is created and validated with QuestionValidationHandler. QuestionValidationHandler checks the question elements and assigns errors and warnings if needed. The errors that can be added are:

- UNKNOWN_QUESTION_TYPE – there is a spelling mistake in the question, or the question type is not supported;
- NO_ANSWER_OPTIONS_FOUND – the answer options either do follow the rules of defining answer options or there are no answer options provided;
- NO_CORRECT_ANSWER_FOUND – none of the answer options are marked as a correct answer;
- INVALID_REGEX – the regex is not recognised as a Java regex pattern.

The added warnings can be:

- CHECK_QUESTION_NAME – the question does not contain '-' or '–', so the whole paragraph is marked as a question name, which is only for Moodle as in Coursera, the question name must always be the number;
- UNKNOWN_PARAGRAPH_TYPE – the paragraph type doesn't match any paragraph rules, so it is skipped;
- ANSWER_OPTIONS_DONT_MATCH_TYPE – the single choice question type has more than one correct answer marked;
- REGEX_RESTRICTION – Moodle and Coursera have their regex restrictions, so there could be some errors when importing the file to Moodle or Coursera due to that.

After the .docx file is processed, the FileUploadService returns a list of Question records containing all the data about the question. The list is filtered, and only the questions without errors are sent to the MoodleXmlCreatorService, where the Moodle XML file is created, or to the CourseraDocxCreatorService where the Coursera .docx is created and returned as a byte array. The unfiltered list is then sent to the QuizDetailsComponent where it is converted into QuizDetails record. The pair of byte array and QuizDetails record is sent back to the controller where the byte array is encoded as base 64 string, FileDto is put together and HTTP request response with FileDto as response body is created (Figure 23).

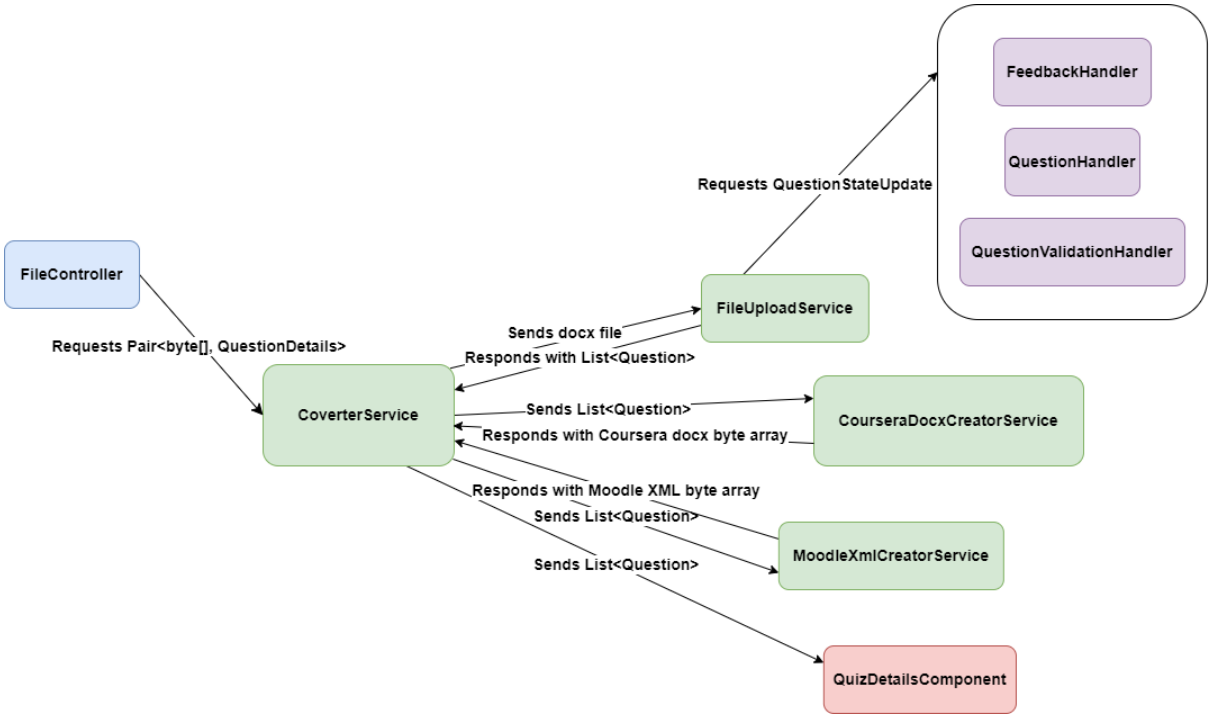


Figure 22. Backend components' relationships.

```
{
    "fileName": "UniTartuCS Template - docs_coursera.docx",
    "file": "UEsDBBQACAgIAGZaqVgAAAAAAAAAAAAAAAATAAAAW0NvbnRlbnRfVHlw
    "details": {
        "questionCount": 2,
        "questionConfigDetails": {
            "SINGLE_CHOICE": 1,
            "MULTIPLE_CHOICE": 1
        },
        "answersCount": 6,
        "questionPicturesCount": 2,
        "answerPictureCount": 0,
        "questionErrors": {
            "Question 4 ": [
                "NO_ANSWER_OPTIONS_FOUND",
                "UNKNOWN_QUESTION_TYPE",
                "NO_CORRECT_ANSWER_FOUND"
            ],
            "Question 5 ": [
                "NO_ANSWER_OPTIONS_FOUND",
                "UNKNOWN_QUESTION_TYPE",
                "NO_CORRECT_ANSWER_FOUND"
            ],
            "Question 3": [
                "NO_ANSWER_OPTIONS_FOUND",
                "UNKNOWN_QUESTION_TYPE",
                "NO_CORRECT_ANSWER_FOUND"
            ]
        },
        "questionWarnings": {
            "Question 4 ": [
                "CHECK_QUESTION_NAME"
            ],
            "Question 5 ": [
                "CHECK_QUESTION_NAME"
            ],
            "Question 3": [
                "CHECK_QUESTION_NAME"
            ]
        },
        "skippedQuestions": 3
    }
}
```


Figure 23. Example of the request response body.

4.4. How frontend works

On the landing page (Figure 24), there is a drop area where users can click to choose the file to upload or drag and drop their .docx file. If the added file does not have .docx extension, then it will not be uploaded, and clicking on Moodle XML or Coursera .docx button will show the warning that there are no files to be uploaded. Once the .docx file is uploaded and the "Moodle XML" or "Coursera .docx" button is clicked, the file is sent to the backend via HTTP request. The frontend will show a small notification that the file is uploading, which means it is waiting for the HTTP response from the backend. Once the response is received, the file is automatically downloaded as an XML or .docx file, and the modal with the file details pops up (Figure 25). The button for opening the modal is shown.

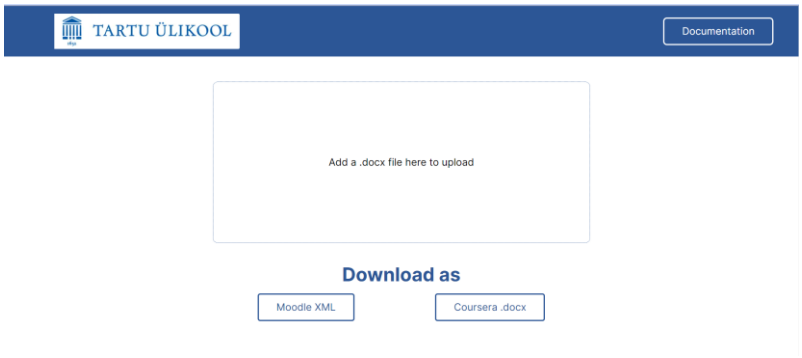


Figure 24. Quiz Converter landing page.

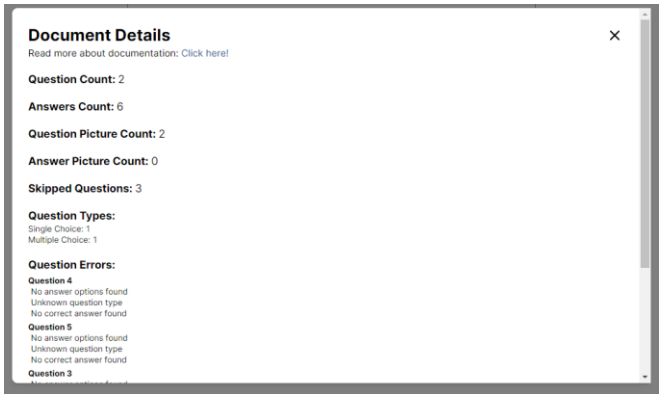


Figure 25. Quiz Converter uploaded document details.

The frontend also has the “Documentation” button, which opens the page with links to the template documentation and the UniTartuCS template (Figure 26).

Figure 26. Quiz Converter documentation page.

4.5. UniTartuCS template

UniTartuCS template (Figure 27) is a .docx template created to make formatting the .docx easier and to avoid mistakes. It has three dropdowns: question type, shuffle option, and credit option. The dropdowns have all the options supported by the Quiz Converter, so choosing from the dropdowns avoids making spelling mistakes or selecting the type that is not supported. The

.docx with dropdowns can be uploaded to the Quiz Converter as the backend supports the dropdowns.

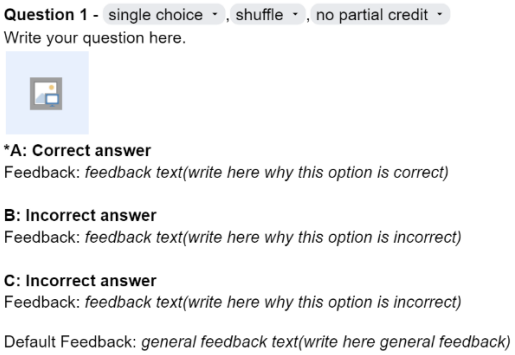


Figure 27. UniTartuCS template single choice question example.

4.6. Implemented features

Table 7 shows implemented features in Quiz Converter. The first column lists the feature, the second column shows if the feature is supported by Coursera and the third column if it is supported by Moodle. Table also shows in its fourth column if the feature was a part of the initial set of requirements or were added during the development phase. The table shows that regular expression question type is only partially supported by Moodle as Moodle does not support regular expression question types by default and therefore the plugin must be installed before using it. The partial credit option is also marked as partially supported by Moodle. That is because Moodle does not support "no partial credit" option. So while the “partial credit” option is supported the “no partial credit” option is not fully supported. The “no partial credit” option is implemented in a way that every incorrect answer automatically gives 0 points to the question. However, if there are three correct and one incorrect answer then choosing one of the three correct answers will give 1/3 of a point.

Table 7. Implemented features in Quiz Convertor.

Feature	**Coursera**	**Moodle**	**Initial requirement**
Allow adding pictures to question description.	Yes	Yes	Yes

Allow adding for pictures to answer options.	Yes	Yes	No
Allow adding feedback to the questions.	Yes	Yes	Yes
Allow adding default feedback to the questions.	Yes	Yes	Yes
Allow single choice question type.	Yes	Yes	Yes
Allow multiple choice question type.	Yes	Yes	Yes
Allow short answer question type.	Yes	Yes	No
Allow regular expression question type.	Yes	Partially*	No
Allow multi-paragraph question descriptions.	Yes	Yes	No
Add automatic file download.	Yes	Yes	Yes
Show information about uploaded file's content.	Yes	Yes	Yes
Allow shuffle/no shuffle questions option.	Yes	Yes	No
Allow partial credit/no partial credit option.	Yes	Partially*	No
Allow getting information from dropdowns from uploaded .docx file.	Yes	Yes	No
Template for creating questions.	Yes	Yes	Yes

Documentation that describes how to use template and what features are supported.	Yes	Yes	Yes

4.7. Deployment

The web application is deployed on an HPC virtual machine with 4 vCPU, 2GB of RAM, and 10 GB disc space. The web application runs in Docker containers: quiz-converter-frontend and quiz-converter-backend.

4.7.1. Docker

According to Docker documentation [42] Docker is a helpful open platform solution used mostly to “box up” software, deliver it between machines and run them. This helps separate the application's dependence on machines and makes it easier to develop and deploy in separate instances or by different developers. As the applications are put into loosely isolated containers, it enables multiple instances of containers to run without meditating too much about applications clashing with each other. Once development is complete, Docker can easily deploy the applications to the final systems. This chapter will overview Docker and its use in this thesis.

4.7.1.1. About Docker

Article by InfoWorld [43] states that Docker was founded in 2010 in a startup incubator. The product was started in France by one of Docker Inc.'s founders, Solomon Hykes. Docker Inc. launched in 2011, and by 2013, it was featured in the PyCon. In the same year, it was released as open source.

Article [43] continuous by bringing out that major firms quickly adopted Docker for their solutions. In 2013, Red Hat adopted Docker for its Red Hat Enterprise Linux, OpenShift and Fedora. Next year, Docker announced a collaboration with Microsoft, Amazon, Stratoscale and IBM for various cloud and server infrastructure projects. As years passed, Docker became more adopted in various systems and infrastructures for everyday users and enterprises. For example, in 2016, Microsoft announced that Docker could be natively used on Windows 10 and in 2019, with the release of WSL and WSL 2, Windows Home users could easily take advantage of the Dockers tools.

4.7.1.2. How Docker works

Docker as a solution could be split into three main parts as illustrated in Figure 28. First, the Docker client, then the host, and finally, the registry. The author will introduce these concepts in the following subchapters and explain how they are used.

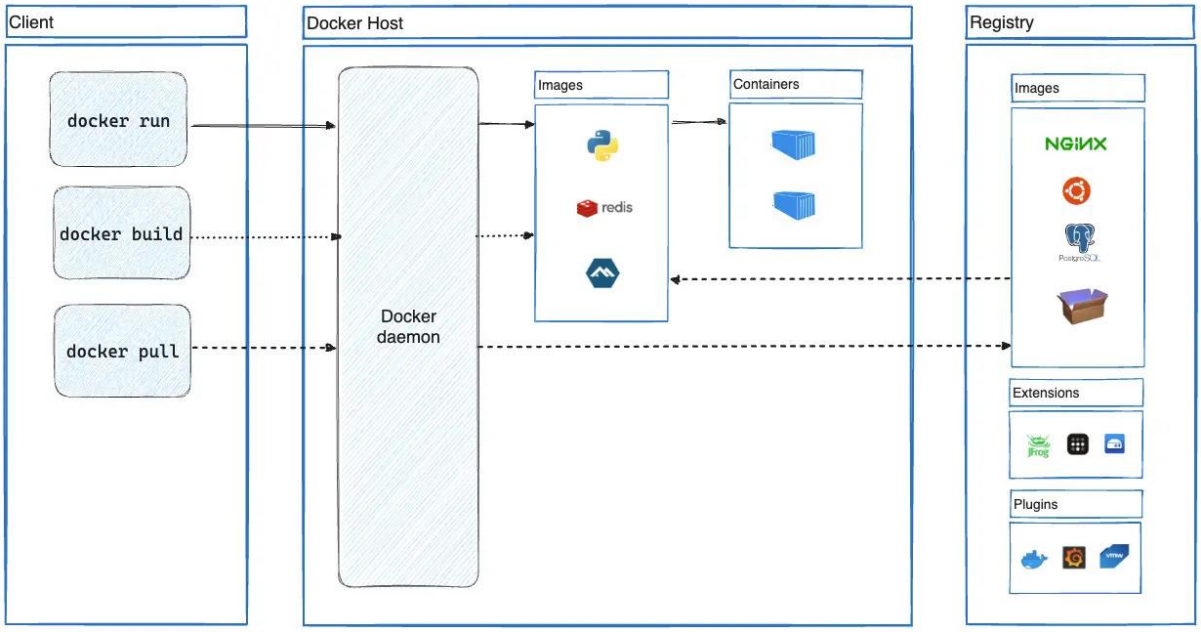


Figure 28. Overview of Docker architecture [42].

4.7.1.3. Docker client

The Docker client according to GeeksforGeeks website [44] is responsible for allowing communication with the underlying Docker ecosystem. It comes in various forms, such as a command line tool or a stand-alone graphical interface called Docker Desktop. Both of these make use of Docker API to communicate with the host, the Docker daemon. The methodology of Docker API relies on REST API as they are essentially the same.

The Docker client has a few use cases:

- Management
    - As Docker mainly uses containers to run different applications in their environments, the Docker client manages the lifecycle of these containers, from creation to deletion.

    - The Docker containers are created using the instructions inside Docker images. As such, the Docker client can be used to push, pull, build, tag or even inspect these images.
    - Once multiple instances of different applications run as containers, Docker Swarm tools can be used to better manage them. The Docker client handles the orchestration of these services and monitors the swarms.
- Monitoring
    - As the Docker is not a standalone solution and needs to borrow resources from its host, monitoring these resources is needed. Docker client keeps watch of available and needed resources like CPU allocation, memory and network usage.
- Development and Testing
    - As developers work on their applications, they can easily use the Docker client to create different environments for their applications. This makes predicting how applications behave in mirrored production environments easier and allows for better coordination between developers.

These use cases show why the Docker client is a highly usable tool, as it helps monitor, test, and manage the Docker host and registry components.

4.7.1.4. Docker host

The Docker host based on Codefresh website [45] does the main lifting behind the scenes. Whilst the Docker client seemed important, it mainly sends REST API requests to the Docker host to query about the states of containers, networks, and images under the host and instructs them on what to do with them. The host must do the main work.

GeeksforGeeks website's article [46] states that a Docker host is essentially a server that runs some operating system and supports Docker containers. It runs the Docker engine, better known as Docker daemon, which mainly isolates containers from one another and acts like a hypervisor, managing and sharing host resources with containers. While this seems to be very similar to how some virtual machine managers and virtual machines work, it is quite different. Whilst every virtual machine runs its own operating system with an underlying kernel, the Docker containers share the Docker host kernel. This means the Docker container includes only the most necessary, like the built application and environmental instructions, allowing

containers to be more compact and easier to manage than the virtual machines themselves. This is visualised in Figure 29. It also means that the physical hardware does not have to be very powerful to run multiple instances of containers, and developers could deploy them easily on their computers.

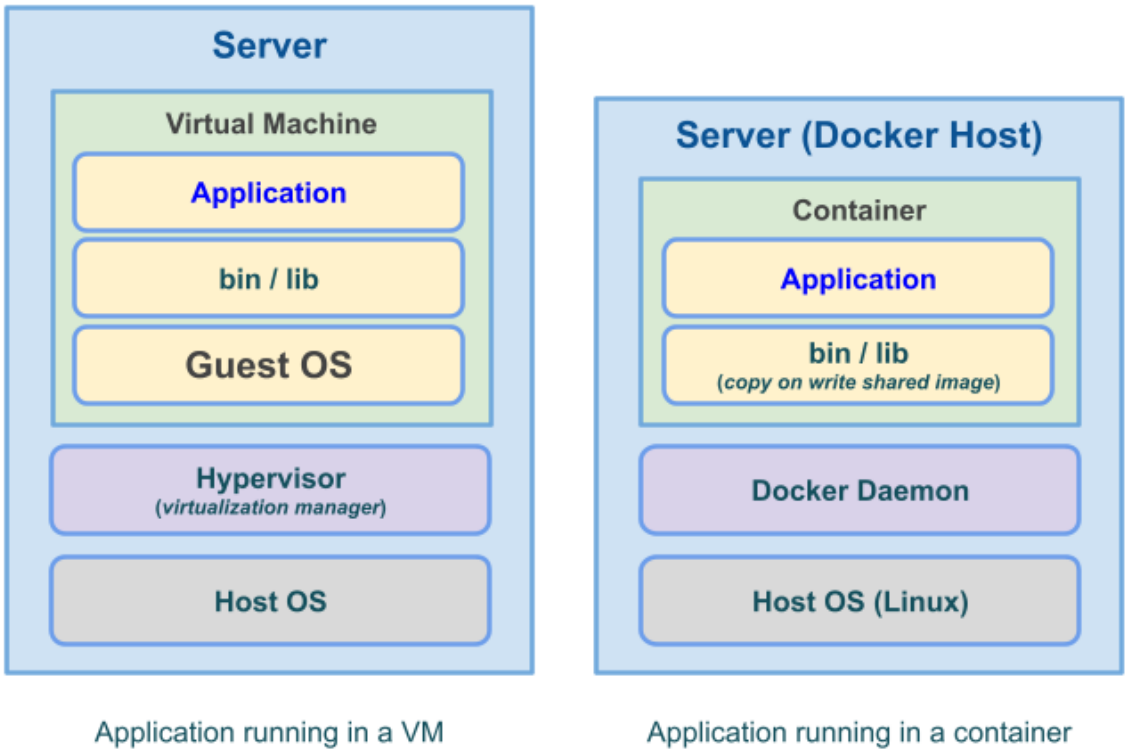


Figure 29. The difference between an application running in a virtual machine vs a Docker container [45].

4.7.1.5. Docker registry

A Docker registry is very similar to a version control system repository or VCS, but at the same time, there are some differences. Based on Atlassian's git tutorial [47] VCS's main purpose is to keep track of document changes in some catalogue tree. It is mainly used for software development purposes where developers may change the same files and have two completely different software source codes simultaneously. This is where VSC helps, allowing users to branch from the main source, implement their changes, and merge them back to the main source. If conflicts arise, they can work together to resolve them; ultimately, they still have one coherent source.

Sysdig's article [48] claims that Docker registry is similar as it allows developers to push their containers; of course, instead of source code, there are compiled applications with instructions

about how to build the necessary environments for them. There is also a second difference in that end consumers can rather easily pull these images from the registry and deploy them more easily than trying to deploy source code from some VCS, assuming that the end user has access to these images. Docker registries mostly eliminate situations where the application runs on one developer's machine but not the others.

4.7.2. Usage in this project

This thesis also makes use of the tools that Docker provides. As mentioned, the application in this thesis consists of two parts: the frontend that serves the user and allows them to upload their quiz templates to be converted, and the backend that receives the template converts it into the required quiz and returns it to the end user with some details. Both the frontend and the backend have a Dockerfile that describes how to build respective applications, what dependencies they have and what their required environments are. For example, Figure 30 has an example of how the backend Dockerfile looks like.

```
FROM maven:3.9.6-eclipse-temurin-21-alpine

RUN mkdir -p /app
WORKDIR /app

COPY pom.xml .
RUN mvn dependency:go-offline -B

COPY . .
RUN mvn package

EXPOSE 8000
CMD ["java", "-jar", "target/converter-0.0.1-SNAPSHOT.jar"]
```

Figure 30. Contents of the thesis's backend Dockerfile.

As seen in Figure 30, the following will be a step-by-step on how Dockerfile instructions will be used to construct applications' backend.

1. The backend needs Maven and Java 21 to run. It takes the described preimage from Docker Hub, a Docker registry hosted by Docker that has bundled both Maven version 3.9.3 and Java version 21 by Eclipse deployed on a small-scale Linux distribution called Alpine.

2. Next, Dockerfile instructs to create a directory called “app” in the root directory of the Linux system. The working directory is moved to the newly created “app” folder.
3. Next, the source code dependencies need to be downloaded. As the backend uses Maven then, its instructor file, “pom.xml”, is moved to the app directory.
4. Next, all the source files of the backend project are copied over, and Maven is executed to build the application.
    a. As seen, Maven dependencies are retrieved from Maven repositories with the additional argument “go-offline”. This means that dependencies are cashed, and if any next Maven commands are run, then they could be run offline, and connection to the World Wide Web is unnecessary.
5. Next, the container is instructed to expose port 8000 to the Docker host as the application backend REST API is configured to run on that port.
6. Finally, the Java run command is used on the compiled Java package, and the application should boot up if no problem arose in the previous steps.

Both the frontend and backend of the application have Dockerfiles with some differences, but the idea remains the same:

1. Retrieve some preimage with bundled necessary runtimes.
2. Copy over the project files
3. Build the application
4. Set up the necessary environment for the application
5. Run the application

The project also has a docker-compose.yml file. This file describes the end Docker host configuration. It contains information:

- About various containers
    - Where to retrieve or how to build them
    - Their environmental attributes set by the host
    - In what order to deploy the containers
- About volumes
    - What volumes are shared by the containers
    - What folder are shared between the host and containers
- About network

- What containers are connected by bridge
- What container networks are isolated

The docker-compose.yml file is a good instruction set and a tool as it enables users to deploy a complex set of containers and their networking configuration in one command. Below, there is an Figure 31 that describes docker-compose.yml written for this thesis application.

```yaml
version: '1'
services:
  frontend:
    container_name: quiz-converter-frontend
    image: quiz-converter-frontend
    depends_on:
      - backend
    build:
      context: ./frontend
      dockerfile: Dockerfile
    ports:
      - 80:3000

  backend:
    container_name: quiz-converter-api
    image: quiz-converter-api
    build:
      context: ./backend
      dockerfile: Dockerfile
    ports:
      - 443:8000
```

Figure 31. Contents of the thesis docker-compose.yml.

As seen in Image C, the purpose is to set up both the applications frontend and back. There are few components to it.

1. Firstly the version in the file describes what version of Docker Compose must be used by the Docker daemon.
2. Next are container descriptions, including frontend and backend.
    a. In frontend instructions, it is stated that it depends on the backend container. This means that Docker Compose will first try to deploy the backend container before trying to deploy the frontend.

b. Both container instructions mention where in the directory the backend and frontend Dockerfiles are located, as will use Dockerfile instructions to build the containers.
c. Both containers also have instructions about what ports must be connected between the Docker daemon and the host machine. This can then be used to access applications from the wider web.

Finally, the application built in this thesis uses a UT HPC virtual machine to host the application. The virtual machine runs Ubuntu 22.04 and contains the application source code and docker-compose.yml to deploy the application. Due to limitations from the virtual machine provided by the UT HPC, the only open ports to the wider web are 80 and 443, which are mostly used to describe HTTP and HTTPS connection channels, respectively. If this application becomes more public, then it would be advised to use proper ports instead. Below is Figure 32, which briefly describes the overall cloud architecture of the application.

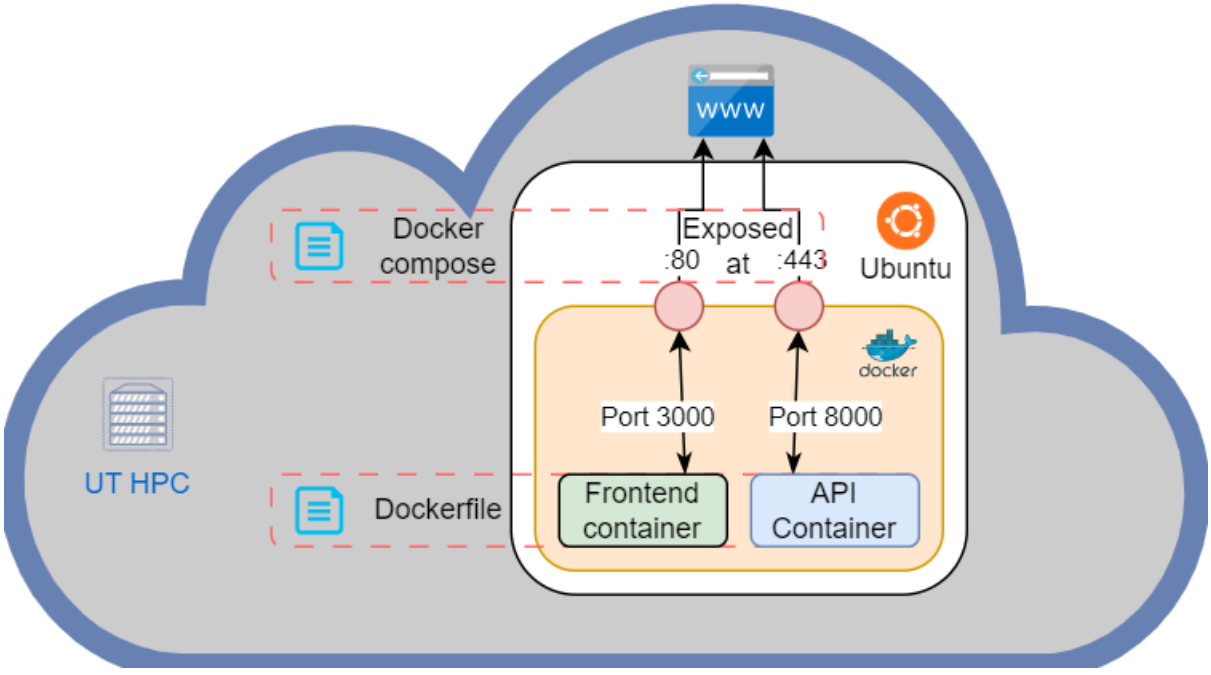


Figure 32. Overview of the applications web architecture.

5. Testing

During the development, each feature was tested manually as it was developed. Each time a new feature related to the quizzes was created, the files with the different versions of related questions were created (Figure 33), converted into Moodle XML and/or Coursera DOCX and then uploaded to the corresponding platform. The imported question would be checked through the LMS view to ensure that it was imported correctly.

Question 21 – text match, shuffle

Eestis elav loom

***A: ilves**

Feedback: õige vastus

B: suitsupääsuke

Feedback: peaagu, küll aga tegu on rahvuslinnuga mitte loomaga

***C: hunt**

Default Feedback: vale vastus

Figure 33. Example of testing short answer question type.

Keeping track of the created test .docx files was getting harder as more features came. Therefore, automated tests were created. In Figure 34 there is an example of a test that tests FileUploadService. The service is created as a variable and named sut, which is an abbreviation of the system under test. The main idea of tests is to import the created .docx files with different question types and see if the FileUploadService gets questions correctly. In the example the test file contains single choice question with an image in the description. The tests checks for correct question type, if it contains picture also for any warnings and errors.

```
public class FileUploadServiceTests {

    FileUploadService sut = new FileUploadService();

    @Test
    void TestGetSingleChoiceWithPictureQuestion() throws IOException {
        MultipartFile multipartFile = new MockMultipartFile( name: "single_choice_picture.docx",
                new FileInputStream(new File( pathname: "src/test/java/resources/single_choice_picture.docx")));
        var questions : List<Question> = sut.convertDocToQuestion(multipartFile);
        var question : Question = questions.get(0);
        Assertions.assertEquals(question.type(), QuestionType.SINGLE_CHOICE);
        Assertions.assertEquals(question.description().getPictures().size(), actual: 1);
        Assertions.assertEquals(question.errors().size(), actual: 0);
        Assertions.assertEquals(question.warnings().size(), actual: 0);
    }
```

Figure 34. Example of the automated test in FileUploadServiceTests file.

6. Conclusion

6.1. Summary

During this Master's thesis, a web application named Quiz Converter was created. The purpose of the web application is to convert DOCX files to the file formats that Coursera and Moodle both accept. The application is aimed at teachers who create questions for their quizzes together. As Moodle and Coursera do not allow commenting on the questions or seeing the history of changes, teachers prefer using Google Drive or Microsoft Cloud to create and keep track of questions. The questions are then manually imported into the learning management system individually. The Quiz Converter automates the last step, minimizing manual work, as the converted DOCX file can be easily imported to Moodle or Coursera platforms.

Moodle and Coursera were studied to understand how creating the questions and quizzes works. As well as what question types exist and what they mean. A comprehensive competitor analysis was carried out to ensure that there are no open-source solutions that already fix the problem.

The interviews were carried out to get a better idea of how teachers create their quizzes. Based on them, the initial requirements for the MVP were defined. These base features were analysed, and the application's UI designs were created. Then, the development technologies that would fit the application needs were chosen.

In order to ensure that the Quiz Converter works manual test were carried out during and after feature development. Later when more features were implemented, the automated tests were added to ensure that new developments would not interfere with already implemented ones.

The application was deployed to the University of Tartu HPC virtual machine on http://193.40.155.50/ and was available for everybody to use on 3th of May 2024.

6.2. Future work

There are still room for development in both feature and management side. In the future more question types could be added to be supported by the Quiz Converter. Ideally it would support all the question types that Moodle and Coursera support (**Error! Reference source not**

found.). Also the support for the question categories and Coursera Assignments could be added. Converting from Moodle XML to the Coursera DOCX and vice versa could be added as well as converting Moodle XML and Coursera DOCX to the Quiz Converter template.

From the management side the application could have continuous integration pipelines that would run automatic tests and deploy the new code versions automatically to the virtual machine. DNS could be introduced to change the application address to quiz-converter.cs.ut, as currently the only way to access the application is thought an IP-address.

References

[1] ‘The Moodle Story - Moodle - Online Education For Everyone’, Moodle. Accessed: May 10, 2024. [Online]. Available: https://moodle.com/about/the-moodle-story/

[2] ‘Online Learning With The World’s Most Popular LMS’, Moodle. Accessed: May 10, 2024. [Online]. Available: https://moodle.com/

[3] ‘Question bank - MoodleDocs’. Accessed: May 11, 2024. [Online]. Available: https://docs.moodle.org/404/en/Question_bank

[4] ‘Building Quiz - MoodleDocs’. Accessed: May 11, 2024. [Online]. Available: https://docs.moodle.org/404/en/Building_Quiz

[5] ‘Coursera’s Mission, Vision, and Commitment to Our Community | Coursera’. Accessed: May 13, 2024. [Online]. Available: https://about.coursera.org/

[6] ‘Introducing the New Assignment Item - February 2023’. Accessed: May 15, 2024. [Online]. Available: https://www.coursera.support/s/article/educator-000001973-introducing-the-new-assignment-item

[7] Iris, ‘Introducing New Tools and Features as Demand for Online Learning Grows’, Coursera Blog. Accessed: May 13, 2024. [Online]. Available: https://blog.coursera.org/introducing-new-platform-innovations-as-demand-for-online-learning-grows/

[8] ‘About Question Banks’. Accessed: May 13, 2024. [Online]. Available: https://www.coursera.support/s/article/360052649072-About-Question-Banks?

[9] ‘Automatically Create Assignment Questions with Assessment Generator’. Accessed: May 13, 2024. [Online]. Available: https://www.coursera.support/s/article/educator-000002179-automatically-create-assignment-questions-with-assessment-generator

[10] ‘Random Short-Answer Matching question type - MoodleDocs’. Accessed: May 15, 2024. [Online]. Available: https://docs.moodle.org/404/en/Random_Short-Answer_Matching_question_type

[11] ‘LMS Migration - K16 Solutions’. Accessed: May 02, 2024. [Online]. Available: https://www.k16solutions.com/product/scaffold-migration/

[12] ‘Import any Word quiz into Canvas, Blackboard or Moodle in one-click’. Accessed: May 02, 2024. [Online]. Available: https://digitaliser.getmarked.ai/

[13] ‘GETMARKED Digitaliser | User Guide’. Accessed: May 02, 2024. [Online]. Available: https://digitaliser.getmarked.ai/guide/

[14] ‘Moodle XML Converter’. Accessed: May 04, 2024. [Online]. Available: https://moodlexml.fly.dev/info/help

[15] M. Huerta and M. A. Fernandez-Ruiz, ‘FastTest Plugin: a New Plugin to Generate Moodle Quiz XML Files’. Preprints, Feb. 22, 2022. doi: 10.20944/preprints202202.0282.v1.

[16] ‘Moodle plugins directory: FastTest PlugIn | Moodle.org’. Accessed: May 02, 2024. [Online]. Available: https://moodle.org/plugins/view.php?id=2831

[17] ‘Moodle plugins directory: Microsoft Word File Import/Export (Question Format)’. Accessed: May 04, 2024. [Online]. Available: https://moodle.org/plugins/qformat_wordtable

[18] ‘Moodle2Word’. Accessed: May 02, 2024. [Online]. Available: http://www.moodle2word.net/

[19] ‘Moodle test creator’. Accessed: May 02, 2024. [Online]. Available: http://text2gift.atwebpages.com/Text2GiftConverter.html

[20] ‘Excel Moodle Integrations | Jordan Svien Excel Tools’, jordan. Accessed: May 02, 2024. [Online]. Available: https://hbubecc.wixsite.com/jordan

[21] ‘PYPL PopularitY of Programming Language index’. Accessed: May 13, 2024. [Online]. Available: https://pypl.github.io/PYPL.html

[22] ‘Next.js Tutorial’, GeeksforGeeks. Accessed: May 15, 2024. [Online]. Available: https://www.geeksforgeeks.org/nextjs/

[23] ‘Pricing information | Aspose.Words for Java’. Accessed: May 05, 2024. [Online]. Available: https://purchase.aspose.com/pricing/words/java/

[24] ‘History of Changes’. Accessed: May 15, 2024. [Online]. Available: https://poi.apache.org/devel/history/changes-pre3x.html

[25] ‘Apache POI - Component Overview’. Accessed: May 15, 2024. [Online]. Available: https://poi.apache.org/components/index.html

[26] ‘Apache POI - HWPF and XWPF - Java API to Handle Microsoft Word Files’. Accessed: May 15, 2024. [Online]. Available: https://poi.apache.org/components/document/index.html

[27] ‘XWPFRun (POI API Documentation)’. Accessed: May 15, 2024. [Online]. Available: https://poi.apache.org/apidocs/dev/org/apache/poi/xwpf/usermodel/XWPFRun.html

[28] ‘Import questions - MoodleDocs’. Accessed: May 15, 2024. [Online]. Available: https://docs.moodle.org/404/en/Import_questions

[29] ‘Aiken format - MoodleDocs’. Accessed: May 15, 2024. [Online]. Available: https://docs.moodle.org/404/en/Aiken_format

[30] ‘GIFT format - MoodleDocs’. Accessed: May 15, 2024. [Online]. Available: https://docs.moodle.org/404/en/GIFT_format

[31] ‘Embedded Answers (Cloze) question type - MoodleDocs’. Accessed: May 15, 2024. [Online]. Available: https://docs.moodle.org/404/en/Embedded_Answers_(Cloze)_question_type

[32] ‘Moodle XML format - MoodleDocs’. Accessed: May 15, 2024. [Online]. Available: https://docs.moodle.org/404/en/Moodle_XML_format

[33] ‘Missing word question format - MoodleDocs’. Accessed: May 15, 2024. [Online]. Available: https://docs.moodle.org/404/en/Missing_word_question_format

[34] ‘Article Detail’. Accessed: May 15, 2024. [Online]. Available: https://www.coursera.support/s/article/360002997431-Import-and-Download-Auto-Graded-Assessment-Questions?

[35] ‘Use YAML and QTI to Import Questions to Quizzes and Assignments’. Accessed: May 15, 2024. [Online]. Available: https://www.coursera.support/s/article/educator-000001646-use-yaml-and-qti-to-import-questions-to-quizzes-and-staff-graded-assignments?

[36] ‘Assessments’. Accessed: May 15, 2024. [Online]. Available: https://www.coursera.support/s/educator-resource-center-assessments

[37] ‘Copy of Example Question Types for Importing on Coursera’, Google Docs. Accessed: May 15, 2024. [Online]. Available: https://docs.google.com/document/d/1VWlNEG6TBgLNKHthCoZsynz6J3WIHjFoEyGjBnnI7Bs/edit?usp=embed_facebook

[38] ‘YAML Tutorial –’. Accessed: May 15, 2024. [Online]. Available: https://rhnh.net/2011/01/31/yaml-tutorial/

[39] ‘Respondus40UserGuideIMS’. Accessed: May 15, 2024. [Online]. Available: https://web.respondus.com/wp-content/uploads/2019/08/Respondus40UserGuideIMS.pdf

[40] ‘Java SE Application Design With MVC’. Accessed: May 07, 2024. [Online]. Available: https://www.oracle.com/technical-resources/articles/javase/mvc.html

[41] ardalis, ‘Overview of ASP.NET Core MVC’. Accessed: May 07, 2024. [Online]. Available: https://learn.microsoft.com/en-us/aspnet/core/mvc/overview?view=aspnetcore-8.0

[42] ‘Docker overview’, Docker Documentation. Accessed: May 09, 2024. [Online]. Available: https://docs.docker.com/get-started/overview/

[43] S. Carey, ‘How Docker broke in half’, InfoWorld. Accessed: May 14, 2024. [Online]. Available: https://www.infoworld.com/article/3632142/how-docker-broke-in-half.html

[44] ‘What Is Docker Client ?’, GeeksforGeeks. Accessed: May 12, 2024. [Online]. Available: https://www.geeksforgeeks.org/what-is-docker-client/

[45] R. Tabib, ‘Docker Machine’, Codefresh. Accessed: May 12, 2024. [Online]. Available: https://codefresh.io/blog/docker-machine-basics/

[46] ‘Docker - Containers & Hosts’, GeeksforGeeks. Accessed: May 12, 2024. [Online]. Available: https://www.geeksforgeeks.org/docker-containers-hosts/

[47] Atlassian, ‘What is version control | Atlassian Git Tutorial’, Atlassian. Accessed: May 12, 2024. [Online]. Available: https://www.atlassian.com/git/tutorials/what-is-version-control

[48] ‘What is a Docker Registry?’, Sysdig. Accessed: May 12, 2024. [Online]. Available: https://sysdig.com/learn-cloud-native/container-security/what-is-a-docker-registry/

Appendix

I. Source Code

Available on https://gitlab.cs.ut.ee/klamas/test-converter

II. UniTartuCS Template

Available on https://docs.google.com/document/d/106xTDS6IG7fofL6Ju4JnqntjxLeBfBM_LIyjKtskFb8/edit?usp=sharing

III. UniTartuCS Template Documentation

Available on https://docs.google.com/document/d/1IZWmkxqYYqBAYqW8uumiZ1k94muKS4IjpYQuOQ8-9WU/edit?usp=sharing

IV. Interview Questions

-Do you use Moodle or Coursera environment to administer the tests?

”Yes” -> How do you manage the tests?

“No” -> For what reason?

-How do you prepare tests?

-Could you describe all the steps from scratch?

-What do you like and dislike?

-Do you cooperate with other lecturers when preparing the tests?

“Yes” -> How do you coordinate cooperation and what tools do you use?

“No” -> Can such a need arise if there are opportunities?

-Do you use a question bank when creating Moodle or Coursera tests?

-How do you manage and select questions from these banks?

-Do you use new technologies or innovative approaches when creating tests?

-What settings do you use when creating tests? (Shuffle, partial credit)

V. License

Non-exclusive licence to reproduce the thesis and make the thesis public

I, Anneli Klamas,

1. grant the University of Tartu a free permit (non-exclusive licence) to
reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright, my thesis

Quiz Converter: The tool for creating quizzes in Coursera and Moodle,

supervised by Jaak Vilo, PhD.

2. I grant the University of Tartu a permit to make the thesis specified in point 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 4.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in points 1 and 2.

4. I confirm that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Anneli Klamas

15/05/2024