BERNARDO MAIA AMARAL

ONLINE INTERACTIVE TOOL FOR LEARNING LOGIC

MASTER IN COMPUTER SCIENCE

NOVA University Lisbon
⟨February⟩, ⟨2022⟩

DEPARTMENT OF
COMPUTER SCIENCE

ONLINE INTERACTIVE TOOL FOR LEARNING LOGIC

BERNARDO MAIA AMARAL

Adviser: Ricardo Gonçalves
Assistant Professor, NOVA University Lisbon

Co-adviser: João Costa Seco
Associate Professor, NOVA University Lisbon

MASTER IN COMPUTER SCIENCE

NOVA University Lisbon
⟨February⟩, ⟨2022⟩

Abstract

Nowadays there are many educational institutions and businesses that have switched their favored training methods to alternative online solutions, such as courses located in Massive Open Online Courses platforms. This process is becoming more and more common for education institutions, and it brings a large number of benefits, namely having a centralized source of content that is easy to keep up to date. Not only that, but the online learning model allows for students to follow their own pace.

In light of this new trend, the purpose of this dissertation is to design and implement an online platform that will be used to solve Logic exercises. This platform will give students that attend logic related courses a tool that will be very useful as a complement to the theoretical classes, such as the students of the University of Nova Lisbon. It will also be used by instructors, as a way to grade students through the Learning Management System (LMS). This problem will bring forth some interesting challenges such as the design of some of the exercises, and the integration of those exercises in the LMS.

For that goal, in this document we overview some related literature about the technologies that are going to be used, as well as give a detailed explanation on how every component of this dissertation will be implemented. We will then thoroughly test our platform and present our results and evaluations.

Keywords: Classical Propositional Logic, Massive Open Online Courses, Learning Management Systems, Online Learning, Learning Tools Interoperability

Resumo

Nos dias de hoje, existem muitas instituições educacionais e negócios que já fizeram uma mudança nos seus métodos de treino favoritos para alternativas como cursos localizados numa plataforma MOOC. Este processo é cada vez mais comum para as instituições educacionais, pois traz um enorme número de benefícios, nomeadamente, fornecendo uma plataforma centralizada com todo o conteúdo, sendo este fácil de manter atualizado. Este modelo de treino online também permite que quem esteja a aprender siga o seu próprio ritmo.

Tendo em conta esta nova tendência, o propósito desta dissertação é criar uma plataforma online para fornecer uma ferramena a estudantes de Lógica, como os da Universidade Nova de Lisboa, para praticar os conteúdos lecionados nas aulas. Esta plataforma também permitirá a instrutores criarem quizzes e assignments no *Learning Management System* (LMS) usando os exercícios implementados para avaliar os estudantes. Este problema irá gerar alguns desafios interessantes, especialmente na criação e desenho dos exercícios, e na integração desses exercícios no LMS.

Para esse objetivo, neste documento nós vamos rever alguma da literatura existente sobre as tecnologias que irão ser usadas para implementar esta plataforma. Vamos também dar uma explicação detalhada sobre como cada componente desta dissertação irá ser implementado. Finalmente, iremos testar rigorosamente a nossa plataforma e apresentar os nossos resultados e avaliações.

Palavras-chave: Classical Propositional Logic, Massive Open Online Courses, Learning Management Systems, Online Learning, Learning Tools Interoperability

Contents

List of Figures

List of Tables

List of Listings

1 Introduction

In this chapter, we introduce the problem we are trying to solve in this dissertation and motivate the necessity of having a custom online platform to be used in a Logic course. Subsequently, we formally define the problem we propose to address in this dissertation, delineate its guiding objectives and highlight the expected contributions. We conclude the chapter with an outline of the document structure.

1.1 Context and Motivation

Lately, there as been an increase in the demand for online courses and alternatives for typical classrooms. For example, following a study of online courses in the United States, it was found that in 2012 there were 1.6 million students who studied at least one online course, and in 2013 and 2014 there were 5.5 million and 7.1 million students, respectively [13], and these numbers keep increasing in more recent times. With this knowledge, many educational institutions are making this transition to provide online versions of their courses. Online learning can satisfy the needs of both active and passive learning, aiming to provide students with complete knowledge that can be accessed anywhere and anytime.

With the growing number of students, having alternative ways to evaluate and grade them automatically without recurring to the usual process of a written test is a major benefit. By simply answering an online quiz that is graded automatically, universities can save massive amounts of time and resources by making this transition to platforms that have this type of functionalities implemented.

Another reason that has caused educational institutions and businesses to convert their training methods to these online platforms is the recent COVID-19 pandemic. This led to a massive increase in the number of people working and studying from home, making it so that these businesses and institutions had a great incentive to convert their training platforms to an online version [21].

In more recent times, we can see the impact and growth of this need of having learning material available online. There are already many existing online platforms that can be used to create Logic courses, and many of them that already have some type of Logic courses available. For example, we can find courses that cover varied Logic subjects in platforms such as EdX and Coursera, and it is possible to create a Moodle course and use the tools available there to create quizzes and assignments. However, to create a course in these platforms it requires the instructor to design the exercises manually and use the tools that these platforms have available. The problem with this is that many of these platforms have limited options for what type of exercises can be added to the course, some mainly utilizing truth or false questions or multiple choices. Also, for each exercise the instructor has to manually set up the grading and feedback individually which makes it a troublesome task whenever there is a need to add new exercises to the platform.

Throughout the planning process of the online platform and the model exercises we wanted to have on it, we soon realized that the existing platforms that already have Logic courses or that allow for the creation of one, were all too generic for the specific requirements we wanted. With this online platform, our main goal is to design model exercises of a given Logic subject, and for each type of exercise, we want to have it be automatically graded and giving immediate feedback to the students while solving them. The model exercises are designed in a way that allows for an instructor to add new exercises of that type very easily, by giving the exercise information in a set format. After uploading this information, the system generates automatically new exercises with the associated solution and feedback. This type of design of the exercises and functionalities associated with it, such as the adding of new exercises through a set language, or the automatic feedback and grading associated to every exercise on the platform, make it so that using an already existing platform, such as Moodle or EdX, not a viable solution.

1.2 Problem Statement and Objectives

Taking into consideration the points that were just made in the previous section, we propose in this dissertation a plan to design and implement an online platform that will have model Logic exercises for learners to practice solving them and integrate those exercises in a Learning Management System (LMS). This platform can be used by students enrolled in a Logic related course, such as those in the University of Nova Lisbon, as a tool to practice the knowledge they acquire in class. It can also be used by instructors to conduct quizzes and assignments through the LMS to grade students.

With that said, we focus on the following main objectives to guide the work of this dissertation:

1. Design and implement an online platform for Logic and its associated auxiliary tools.

2. Integrate the previously implemented platform in a course from a Learning Management System (LMS), allowing instructors to grade students automatically.

3. Provide instructors with an easy way to add new exercises (of a given model exercise) by posting the exercise information in a specific language/format.
4. Every type of model exercise in this platform will be automatically graded and give immediate feedback to students.

1.3 Expected Contributions

By addressing the problem and objectives just discussed in 1.2, we expect to achieve the following contributions:

- Establishing a training platform for students of Logic to practice their knowledge obtained in class.
- Creating a course on a Learning Management System that utilizes the tools and exercises that will be implemented for this dissertation to grade students.
- Developing a comprehensive guide on how to establish this connection between outside tools and LMS.

1.4 Document Structure

This document is structured as follows. In chapter 2, we give some background on the concepts and topics of Logic that will be used in this dissertation, as well as some context about massive online courses. In chapter 3, we give an overview on the architecture of the system, details on the exercises that will be on the platform and discuss the logic oriented tools that are going to be implemented for this dissertation. Afterwards, we delineate the work plan according to the objectives and conclude with the task schedule.

2 Background

In this chapter we will talk about a relatively recent development in distance learning, called Massive Open Online Courses (MOOCs), in section 2.1. We will discuss what they are, some of their advantages and disadvantages and we will establish a relationship between some of the most well known MOOCs and the platform that is going to be created in this dissertation. To conclude the chapter we will give some background on the concepts of Logic that will be used in the design of the exercises, in section 2.2.

2.1 Massive Open Online Courses

Massive Open Online Courses (MOOCs) are free online courses available for anyone to enroll. They provide an affordable and flexible way to learn new skills, by allowing millions of people around the world to learn for a variety of reasons, including: career development, changing careers, college preparations, supplemental learning, lifelong learning, corporate eLearning & training, and more. MOOCs share several key features, the most obvious one is that all content is delivered online, either through video, slideshows, discussion boards, or any combination thereof.

The growing popularity of MOOCs can be attributed to a number of factors. Increasing access to the internet and increased bandwidth has made MOOCs accessible to students from many geographically distributed places, and has created opportunities for online interaction within a global cohort. For computer science subjects, MOOCs offer the potential to lower course costs and provide accessibility to large number of students that wouldn't have this type of opportunities otherwise [18].

2.1.1 Advantages and Disadvantages

There are many advantages to the MOOC model for online education. The inherent openness and user-friendliness of the format means that incredible educational resources are available to anyone with the time and money to devote to learning, offering a real opportunity to people without access to traditional education. It allows instructors to divide the lectured content of the courses into modules giving it more flexibility for learners to find the information they need. These courses also bring a very easy way to keep information centralized and up to date to make the learners job easier when searching for this type of content [14].

MOOCs are all about learning at your own pace. The learners can log on to the portal whenever they wish and access the course. The development of MOOCs is made possible by the online exchange of information by experts through social networking platforms. However, there are still many issues that remain unresolved. MOOCs are often touted as a way of teaching thousands of students at once, but the reality is that very few students who register for a MOOC complete it. Many never even begin the course after registering. A particular study made on this problem found that MOOCs could have a completion rate of less than 5% in some cases [8]. Even when students do complete their work, there is no guarantee they have successfully mastered the material. Although the idea of cheap and available education for everyone is noble, in practice few students have the perseverance to work their way through the material without additional motivation.

Based on statistic study result and the dropout prediction system experiments result, several suggestions are made to help improve course management in the perspective of dropout prevention, such as offering students more chance to try quizzes and assignments, prolong the period of accomplishing graded assignments, encouraging students to participate in forum discussion and designing in-video quizzes to divide each video into short fragments [4].

2.1.2 Alternative Solutions to the Problem

In this section we will discuss some of the massive open online courses available in the market, and in the end compare them with the platform we will implement in this dissertation.

2.1.2.1 MOOCs options

Here we list some popular options MOOC platforms.

1. **edX**

   edX [7] was founded in 2012 by a partnership between MIT and Harvard. It hosts online university-level courses in a wide range of disciplines to a worldwide student body, including some courses at no charge. Its purpose is to share the school's quality educational resources and provide students with interactive online learning courses. The development goal of edX is to establish a global MOOCs.

   It is of great significance to study teaching methods and explore the issues of online and offline mixed teaching modes, educational effectiveness evaluation, teaching methods, distance education effects, and academic management. edX's assessment model is a combination of online testing and offline examination center testing. After the course passes, a certificate of completion will be issued. The online courses provided by edX can not only be used in conjunction with on-campus teaching, but also provide global learners with quality educational resources.

2. **Coursera**

   Coursera [5] was founded in March 2012 by two computer science professors at Stanford University. The course covers many subjects including computer science, mathematics, and others. With more than 4,000 classes, it is the platform with the largest number of courses [6]. In terms of assessment, Coursera provides online quizzes, assignments and exercises. It also designs a distinctive peer review system for learners. By training learners to use the scoring rules to correct and assess classmates' assignments. It enables learners to obtain more accurate feedback and also gains learning experience in the process of peer review.

   The teaching videos in Coursera are mainly divided into three categories: course introduction, teaching content, and exercise explanation. Through the collection and classification of questions in the forum, teachers can purposely pre-record and upload Q&A videos. Each lesson is broken down into a number of related topic videos to facilitate learners to learn by topic and to use spare time for fragmented learning. Most courses provide both video and text learning materials. While watching the course video, the text data can help learners follow the teacher's thinking to learn [15].

2.1.2.2 Iltis: Learning Logic in the Web

The final platform that we are going to discuss in this section, before comparing them with our dissertation, is a project called Iltis, created by the TU Dortmund University and associates [20].

Out of all the platforms mentioned, Iltis is the one more closely related to the work we are planning to do on this dissertation. It is also a platform that focuses on the discipline of Logic, giving an experience similar to a typical course on a MOOC. For each subject of Logic, Iltis contains text and images explaining the topics and concepts, and then the learners can proceed to solve exercises and do practical tests, to get a general idea of how their understanding of the subject in question is.

Many of the exercises and overall design of the platform served as inspiration for some of the ideas of our dissertation, with some of the exercises we are going to design having similarities to the Iltis platform. The key difference is Iltis is a static web page that resembles a course from a traditional MOOC, with exercises to practice and with complementary texts that explain the subjects and concepts. Our platform is aimed more to be used as a tool to complement the classes of Logic in the University of Nova Lisbon, to allow students to focus on a more practical approach to add to the experiences gathered in the theoretical classes.

As we mentioned, Iltis is a static web page with fixed content. While this allows a learner to have a brief but complete run through the subjects, it has a very limited number of exercises a student can solve. Our platform focuses on a key aspect - design and implement the system in a way that instructors can easily add different exercises with new data, to each type of exercise implemented. An instructor simply as to load a file with the exercise information structured in such a way that the system knows how to read it, and from that data proceed to then generate a new exercise. This exercises are immediately available to learners as soon as the instructor uploads them, making it easy for instructors to give material for students to prepare to written tests, or graded assignments on the LMS platform.

2.1.3 Comparing Our Plan With the Current Solutions Available

While most massive open online courses are platforms for instructors to post the courses they made for their students to learn about a given subject and also practice working on it, the main objective of our platform is to be used in the classes of Logic from the University of Nova Lisbon. Our platform mainly has exercises that explore several topics of Logic, so that a learner can go there after class and practice what he learned that day. All the exercises are thoroughly explained and the student is given feedback every time he gets something wrong. This method was picked because it helps the student consolidate his knowledge, and understand how to solve certain problems, after learning from the feedback.

MOOCs in general have a hard time keeping students interested and the completion rates of courses on these platforms is extremely low. This is due to the lack of motivation from the students to complete the courses and is mainly caused because these platforms fail to keep students interested. As our platform isn't made to upload entire courses, it focuses instead only on the practical part of the Logic classes, which includes solving exercises, implementing algorithms and understanding the practical aspects of Logic. All of this is meant to be done as the student attends the theoretical classes, to provide him with a better knowledge of the subjects learned in class.

2.2 Logic

In this section we will give some context on the subjects of Logic that are going to be studied in this dissertation [3]. We will mention some of the basic concepts and notions of **Classical Propositional Logic** in 2.2.1, along side with some more complex topics that will be further explored in some of the exercises of the platform - a detailed explanation on each of the exercises that will be in our platform can be found on chapter 3.2.

2.2.1 Classical Propositional Logic

Propositional logic is a branch of logic, philosophy, and discrete mathematics that focuses on the study of propositions and their relationships. The discipline was developed for the purpose of formalizing logical reasoning over formal disciplines such as mathematics and was further extended into computing.

A proposition is the basic building block of logic. It is defined as a declarative sentence that is either True or False, but not both. The Truth Value of a proposition is True(denoted as T) if it is a true statement, and False(denoted as F) if it is a false statement. To represent propositions we use propositional symbols. By convention, these symbols are represented by lower-case roman letters such as p, q, r, s. An atomic proposition is one whose truth or falsity does not depend on the truth-value of any other proposition. It is also possible to construct new propositions, using one or more propositions and joining them with logical connectives, called compound propositions. A connective of PL is any of the symbols '$\neg$', '$\wedge$', '$\vee$', '$\rightarrow$', and '$\leftrightarrow$'. A literal is simply a propositional symbol or its negation.

In classical propositional logic, a propositional formula is often more briefly referred to as a "proposition", but, more precisely, a propositional formula is not a proposition but a formal expression that denotes a proposition, which needs to follow a set of rules to make sure they are well-formed. A well-formed formula (hereafter abbreviated as wff) of Propositional Logic is defined inductively as follows:

1. Any propositional symbol is a well-formed formula.
2. If α is a well-formed formula, then so is $\neg\alpha$.
3. If α and β are well-formed formulas, then so is $(\alpha \wedge \beta)$.
4. If α and β are well-formed formulas, then so is $(\alpha \vee \beta)$.
5. If α and β are well-formed formulas, then so is $(\alpha \rightarrow \beta)$.
6. If α and β are well-formed formulas, then so is $(\alpha \leftrightarrow \beta)$.
7. Nothing that cannot be constructed by successive steps of (1)-(6) is a well-formed formula.

Now that we have given a little bit of the required knowledge to understand the syntax of a Classical Propositional Logic formula, we will now provide a meaning or semantics to said formulas. Our definition will be following the inductive definition of syntax, that was just mentioned right above this paragraph. The semantics of formulas in a logic, are typically defined with respect to a model, which identifies a "world" in which certain facts are true. In the case of classical propositional logic, this world or model is a truth valuation or assignment that assigns a truth value (true/false) to every proposition.

A proposition is built from atomic propositions joined together with the logical connectives, meaning the truth or falsity of said proposition depends on the truth-value of its components. For example, the compound statement $p \to (q \vee \neg r)$ is built using the logical connectives $\to$, $\vee$, and $\neg$. The truth or falsity of $p \to (q \vee \neg r)$ depends on the truth or falsity of p, q, and s.

The logical connectives are also known as truth-functional connectives. These connectives are called "truth functional" because the truth value of a complex proposition built up using these connectives depends on nothing more than the truth values of the simpler propositions from which it is built. Because of this, we can explain the meaning of a truth-functional connective in a couple of ways, the easiest one being through the construction of a truth table. A truth table that shows how the truth value of a proposition formed with the connective depends on the truth values of the proposition's immediate parts. We will give such tables for each of the connectives we introduce in the coming paragraphs.

The symbols '$\neg$', '$\wedge$', '$\vee$', '$\to$', and '$\leftrightarrow$', correspond, respectively, to the truth-functions of negation, conjunction, disjunction, implication and equivalence. We shall consider these individually.

Negation: The negation of a proposition α, simply written $\neg\alpha$ in Propositional Logic, is regarded as true if α is false, and false if α is true. The corresponding table can therefore represent the truth-function of the negation connective:

α	$\neg\alpha$
T	F
F	T

Figure 2.1: Truth table for the negation of a proposition

Conjunction: The conjunction of two propositions α and β, written in Propositional Logic as $(\alpha \wedge \beta)$, is true if both α and β are true, and is false if either α or β is false or both are false. In effect, the meaning of the operator '$\wedge$' can be displayed according to the following table, which shows the truth-value of the conjunction depending on all the values of the propositional symbols:

α	β	$(\alpha \wedge \beta)$
T	T	T
T	F	F
F	T	F
F	F	F

Figure 2.2: Truth table for the conjunction of two propositions

In a proposition of the form $(\alpha \wedge \beta)$, the two propositions joined together α and β, are called the conjuncts, and the whole proposition is called a conjunction.

Disjunction: The disjunction of two propositions α and β, written in Propositional Logic as $(\alpha \vee \beta)$, is true if either α is true or β is true, or both α and β are true; and is false only if both α and β are false. A table similar to the one given above for the conjunction connective, modified for to show the meaning of the disjunction symbol '$\vee$' instead, would be drawn as follows:

α	β	$(\alpha \vee \beta)$
T	T	T
T	F	T
F	T	T
F	F	F

Figure 2.3: Truth table for the disjunction of two propositions

In a proposition of the form $(\alpha \vee \beta)$, the two propositions joined together α and β, are called the disjuncts, and the whole proposition is called a disjunction.

Implication: This logical connective is represented in Propositional Logic with the symbol '$\rightarrow$'. A proposition of the form $(\alpha \rightarrow \beta)$ is false, if α is true and β is false; and is true if either α is false or β is true (or both). This implication generates the following truth-table

α	β	$(\alpha \rightarrow \beta)$
T	T	T
T	F	F
F	T	T
F	F	T

Figure 2.4: Truth table for the implication of a proposition

In a proposition of the form $(\alpha \rightarrow \beta)$, we call α the antecedent, and we call β the consequent, and the whole proposition $(\alpha \rightarrow \beta)$ is sometimes also called a conditional.

Equivalence: This logical connective is represented in Propositional Logic with the symbol '$\leftrightarrow$'. A proposition of the form $(\alpha \leftrightarrow \beta)$ is regarded as true if α and β are either both true or both false, and is regarded as false if they have different truth-values. Hence, we have the following table:

Now that we have a more basic understanding of what are propositional formulas, we will start introducing other important concepts necessary for the comprehension of most of the exercises, beginning with tautologies and logical equivalences.

In Propositional Logic, a well-formed formula is a tautology, if it is the case that for every possible truth-value assignment of the propositional symbols of a formula, that formula has always the truth-value of 1 or T("True"). Tautologies are also sometimes

α	β	$(\alpha \leftrightarrow \beta)$
T	T	T
T	F	F
F	T	F
F	F	T

Figure 2.5: Truth table for the equivalence of two propositions

called logical truths because tautologies can be recognized as true solely in virtue of the principles of propositional logic, and without recourse to any additional information.

In contrast to the definition of tautology, a well-formed formula that for every possible truth-value assignment of the propositional symbols of the formula, has the truth-value of 0 or F("False"), that proposition is called a contradiction. A proposition that is neither contradictory nor tautological is called a possible proposition. A possible proposition is true for some truth-value assignments to its propositional symbols and false for others.

Here are some examples of the definitions we just gave:

- $\alpha \vee \neg\alpha$ is a **tautology**.
- $\alpha \wedge \neg\alpha$ is a **contradiction**.
- $\alpha \vee \beta$ is **possible**.

A different way of proving that two propositions are also logically equivalent consists in using truth tables - both truth table of each given formula must be identical for all combinations of propositional symbols. This method is not always feasible since the propositions can be increasingly complex both in the number of propositional variables used and size of the expression.

Tautologies in logic can also be used to show that two propositions are logically equivalent. In propositional logic in particular, α and β are equivalent propositions, if the compound proposition $\alpha \leftrightarrow \beta$ is a tautology. The notions of equivalence and tautology are very useful since they allow proving that propositions or theorems with different **syntax** can have in fact the same **semantics**.

We will now list some important equivalence transformations:

1. **Identity Laws**
   - $\alpha \wedge T \equiv \alpha$
   - $\alpha \vee F \equiv \alpha$
2. **Domination Laws**
   - $\alpha \wedge F \equiv F$
   - $\alpha \vee T \equiv T$
3. **Idempotent Laws**
   - $\alpha \wedge \alpha \equiv \alpha$
   - $\alpha \vee \alpha \equiv \alpha$
4. **Double Negation Laws**
   - $\neg(\neg\alpha) \equiv \alpha$

5. **Commutative Laws**
   - $\alpha \land \beta \equiv \beta \land \alpha$
   - $\alpha \lor \beta \equiv \beta \lor \alpha$
6. **Associative Laws**
   - $(\alpha \land \beta) \land \gamma \equiv \alpha \land (\beta \land \gamma)$
   - $(\alpha \lor \beta) \lor \gamma \equiv \alpha \lor (\beta \lor \gamma)$
7. **Distributive Laws**
   - $\alpha \land (\beta \lor \gamma) \equiv (\alpha \land \beta) \lor (\alpha \land \gamma)$
   - $\alpha \lor (\beta \land \gamma) \equiv (\alpha \lor \beta) \land (\alpha \lor \gamma)$
8. **De Morgan's Laws**
   - $\neg(\alpha \land \beta) \equiv (\neg\alpha) \lor (\neg\beta)$
   - $\neg(\alpha \lor \beta) \equiv (\neg\alpha) \land (\neg\beta)$
9. **Absorption Laws**
   - $\alpha \land (\alpha \lor \beta) \equiv \alpha$
   - $\alpha \lor (\alpha \land \beta) \equiv \alpha$
10. **Negation Laws**
    - $\alpha \land \neg\alpha \equiv F$
    - $\alpha \lor \neg\alpha \equiv T$
11. **Implication Laws**
    - $\alpha \to \beta \equiv \neg\alpha \lor \beta$
    - $\alpha \to \beta \equiv \neg\beta \to \neg\alpha$
    - $\alpha \land \beta \equiv \neg(\beta \to \neg\alpha)$
12. **Conjunction of Implications**
    - $(\alpha \to \beta) \land (\alpha \to \gamma) \equiv \alpha \to (\beta \land \gamma)$
    - $(\alpha \to \gamma) \land (\beta \to \gamma) \equiv (\alpha \lor \beta \to \gamma)$
13. **Disjunction of Implications**
    - $(\alpha \to \beta) \lor (\alpha \to \gamma) \equiv \alpha \to (\beta \lor \gamma)$
    - $(\alpha \to \gamma) \lor (\beta \to \gamma) \equiv (\alpha \land \beta \to \gamma)$
14. **Equivalence Laws**
    - $\alpha \leftrightarrow \beta \equiv (\alpha \to \beta) \land (\beta \to \alpha)$
    - $\alpha \leftrightarrow \beta \equiv \neg\alpha \leftrightarrow \neg\beta$
    - $\alpha \leftrightarrow \beta \equiv (\alpha \land \beta) \lor (\neg\alpha \land \neg\beta)$
    - $\neg(\alpha \leftrightarrow \beta) \equiv \alpha \leftrightarrow \neg\beta$

Now that we know what two propositions being equivalent means, and how we can transform one proposition into another one that is logically equivalent, we will now talk about normal forms, starting with the conjunctive and disjunctive normal forms. To understand these concepts, we need to know what conjunctive and disjunctive clauses are: a conjunctive clause only contains propositional symbols connected with the $\land$ operator, and the disjunctive clause only contains propositional symbols connected with the $\lor$ operator - in both cases, the symbols can be negated, but only when directly applied to a symbol and not the entire clause. When we join multiple disjunctive clauses together with the $\land$ operator, we have what is called a **conjunctive normal form**. Oppositely, if we have many conjunctive clauses joined together with a $\lor$ operator, we have what is called a **disjunctive normal form**.

Additionally to the previously discussed normal forms, there is also a third type of normal form that is weaker than the two previous normal forms that were described above, called **negation normal form**. A formula is in NNF if and only if the negation operator ($\neg$) only occurs directly in front of a propositional symbol. This normal form is said to be weaker than the previous two, because a formula can be in the negation normal form, but not necessarily in the conjunctive or disjunctive normal form. But in the opposite case, where a formula is in conjunctive or disjunctive normal form, it also is in the negation normal form.

Here are some examples of normal forms:

- $(\alpha \wedge \beta \wedge \neg\gamma \wedge \theta) \vee (\neg\beta \wedge \theta) \vee (\alpha \wedge \theta)$ is in disjunctive normal form.
- $(\alpha \vee \beta \vee \neg\gamma \vee \theta) \wedge (\neg\beta \vee \theta) \wedge \neg\theta$ is in conjunctive normal form.
- $\neg\alpha \wedge \neg\beta$ is in negation normal form.
- $\neg(\alpha \vee \beta)$ is not in negation normal form.
- $(\alpha \wedge (\neg\beta \vee \gamma) \wedge \neg\gamma) \vee \theta$ is in negation normal form but isn't neither in conjunctive normal form or disjunctive normal form.

Now we are going to talk about the topic of satisfiability. A compound proposition α is satisfiable if there is a truth assignment that satisfies α; that is, at least one entry of its truth table is true. A compound proposition α is unsatisfiable (or a contradiction) if it is not satisfiable; that is, all entries of its truth table are false. The propositional satisfiability problem (often called SAT) is the problem of determining whether a set of sentences in Propositional Logic is satisfiable. The problem is significant both because the question of satisfiability is important in its own right and because many other questions in Propositional Logic can be reduced to that of propositional satisfiability.

One of the ways of testing satisfiability is with the use of the HORNSAT algorithm. In formal logic, Horn-satisfiability, or HORNSAT, is the problem of deciding whether a given set of propositional Horn clauses is satisfiable or not. A Horn clause is a clause with at most one positive literal, called the head of the clause, and any number of negative literals, forming the body of the clause. Any formula formed by a conjunction of Horn clauses is called a Horn Formula.

The problem of Horn satisfiability can be solved in polynomial time. A polynomial-time algorithm for Horn satisfiability is based on the rule of **unit propagation**: if the formula contains a clause composed of a single literal p (unit clause), then all clauses containing p (except the unit clause itself) are removed, and all clauses containing $\neg p$ have this literal removed. The result of the second rule may itself be a unit clause, which is propagated in the same manner. If there are no unit clauses, the formula can be satisfied by simply setting all remaining variables negative. The formula is unsatisfiable if this transformation generates a pair of opposite unit clauses p and $\neg p$.

Listing 2.1: HornSAT Algorithm

```
function HORN(θ):
begin function
    mark all occurrences of T in θ;
    while there is a conjunct $P1 \wedge P2 \wedge ... \wedge Pk \rightarrow$ P' of $\theta$ do
        mark P'
    end while
    if $\perp$ is marked then return 'unsat' else return 'sat'
end function
```

This algorithm also allows determining a truth assignment of satisfiable Horn formulas: all propositional symbols contained in a unit clause are set to the value satisfying that unit clause; all other literals are set to false. The resulting assignment is the minimal model of the Horn formula, that is, the assignment having a minimal set of variables assigned to true, where comparison is made using set containment.

When using this algorithm, we need to note that it only works with Horn formulas, meaning that if we want to prove a propositional formula is satisfiable, we first need to make sure that formula is in the form of a conjunction of Horn clauses, by applying the equivalence transformations that we talked about previously on the original formula.

In contrast with the previous HORNSAT algorithm, that showed us if a formula was satisfiable or not, the **resolution rule** is used to prove that a given formula is **not satisfiable**, or **unsatisfiable**. In mathematical logic, resolution is a rule of inference leading to a refutation complete theorem-proving technique for sentences in propositional logic and first-order logic. A clause produced by a resolution rule is sometimes called a resolvent.

The resolution rule in propositional logic is a single valid inference rule that produces a new clause implied by two clauses containing complementary literals (propositional symbol or the negation of a propositional symbol). Two literals are said to be complements if one is the negation of the other (in the following, p is taken to be the complement to $\neg p$.

To better understand the resolution rule, suppose we have the following clauses [p,q] and [$\neg q$,r]. In the first clause we know that either p or q is True. In the second clause we know that q is False or r is True. Both clauses have a complementary literal (q), and we can conclude that, if q is False, then by the first clause p is True; if q is True, by the second clause we can assume r is True. Since q has to have the truth-value of True or False, then it must be the case that either p or r is True. Following this thought process, we are able to derive the clause [p,r]. The intuition for this rule of is pretty simple: given a clause containing a literal p and another clause containing the literal $\neg p$, we can infer the clause consisting of all the literals of both clauses without the complementary pair.

For the case of resolution in prepositional logic, the rule can only be applied to a propositional formula in the conjunctive normal form, to allow the creation of the set of clauses to start applying the rule. There is always the possibility of applying equivalence transformations to a given formula, and apply the resolution rule in the resulting formula, if it is in the conjunctive normal form. If we manage to derive an empty set, this means the original formula was unsatisfiable, as we wanted to show. In the case where we cant derive the empty set, we can't reach the conclusion that the given formula is unsatisfiable.

3 Proposed Work

In this chapter we will give an overview on the proposed work for this project. Firstly, in section 3.1, we discuss the technologies that will be used to implement this online platform for logic and take a look into the architecture of the system, explaining the thought process behind the choices made along the way. Following the technologies section, we discuss some of the possible exercises that will be added onto the platform in section 3.2, giving details on the requirements of each exercise and on the feedback given to the user. Lastly, in section 3.3, we talk about some of the logic oriented tools that will be implemented to help with the execution and implementation of some exercises.

3.1 Technologies

In this section we will present all the technologies that are going to be used to develop and implement this online platform. Firstly, we are going to take a look at the architecture of our platform, giving a visual representation of how it is going to work, and describing the choices made for the programming languages and tools used. To end this section, we are going to describe what learning management systems are and how they are going to be a fundamental part of this project. After that we explore some of our options, comparing the characteristics of two different LMS - Moodle and openEdX

3.1.1 Architecture

In this section we will describe what type of system architecture we will use for our project and give some insights on the choices of the technologies that were made. Starting the section, we present figure 3.1 that shows how the different layers interact with each other, and after that, for each of the components, we discuss the used programming languages and frameworks that will be used in the implementation phase of this dissertation.

The architecture model that we will be using is called a three-tier architecture - this architecture is a well-established software application architecture that organizes applications into three logical and physical computing tiers: the presentation tier, or user interface; the application tier, where data is processed; and the data tier, where the data

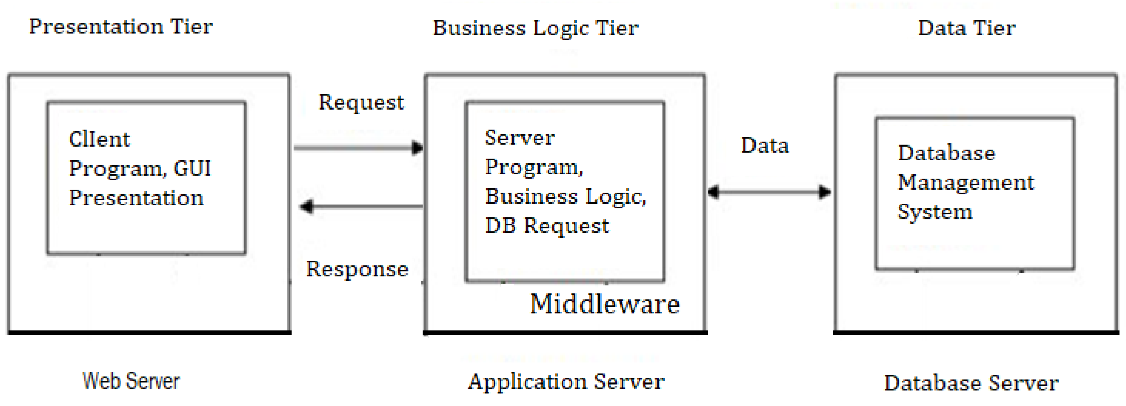


Figure 3.1: Systems architecture

associated with the application is stored and managed. The main benefit of a three-tier architecture is its logical and physical separation of functionality. Each tier can run on a separate operating system and server platform - e.g., web server, application server, database server - that best fits its functional requirements. And each tier runs on at least one dedicated server hardware or virtual server, so the services of each tier can be customized and optimized without impact the other tiers [9].

To list some of the benefits of using a three-tier architecture, we have the following:

- **Compartmentalized development:** Because each tier can be developed independently, we can use the latest tools and best languages for each tier, as well as have things separated for a more organized work experience.
- **Improved scalability:** Any tier can be scaled independently of the others as needed.
- **Improved reliability:** An outage in one tier is less likely to impact the availability or performance of the other tiers.
- **Improved security:** Because the presentation tier and data tier can't communicate directly, a well-designed application tier can function as a sort of internal firewall, preventing SQL injections and other malicious exploits.

3.1.1.1 Presentation Tier - Web Application

The presentation tier is the user interface and communication layer of the application, where the end user interacts with the application. Its main purpose is to display information to and collect information from the user. This top-level tier can run on a web browser, as desktop application, or a graphical user interface (GUI), for example.

For this layer, React is going to be used to implement the front-end of our platform. React allows developers to create large web applications that can change data, without reloading the page. The main purpose of React is to be fast, scalable, and simple. In addition, React's community is very large and there are a lot of supporting materials out there to complement our work.

React React is a free and open-source front-end JavaScript library for building user interfaces based on UI components. It is maintained by Meta (formerly Facebook) and a community of individual developers and companies. React can be used as a base in the development of single-page or mobile applications. However, React is only concerned with state management and rendering that state to the DOM, so creating React applications usually requires the use of additional libraries for routing, as well as certain client-side functionality.

3.1.1.2 Application Tier - Server

The application tier, also known as the logic tier or middle tier, is the heart of the application. In this tier, information collected in the presentation tier is processed - sometimes against other information in the data tier - using business logic, a specific set of business rules. The application tier can also add, delete or modify data in the data tier.

For this layer, a choice still hasn't been made on which programming language will be used. During the implementation phase of the dissertation, we will explore some of our options and decide what language to use then.

3.1.1.3 Data Tier - Database

The data tier, sometimes called database tier, data access tier or back-end, is where the information processed by the application is stored and managed.

We are going to use MySQL, because it is one of the databases that are compatible with Moodle. Another valid reason as to why we chose MySQL is because for our type of data, mainly the exercises, a relational data model allows us to organize our data in a very efficient way.

MySQL MySQL is an open-source relational database management system (RDBMS). A relational database organizes data into one or more data tables in which data types may be related to each other; these relations help structure the data. SQL is a language programmers use to create, modify and extract data from the relational database, as well as control user access to the database. In addition to relational databases and SQL, an RDBMS like MySQL works with an operating system to implement a relational database in a computer's storage system, manages users, allows for network access and facilitates testing database integrity and creation of backups.

3.1.2 Learning Management System

A learning management system (LMS) is a software application or web-based technology used to plan, implement, and assess a specific learning process. Typically, a learning management system provides an instructor with a way to create and deliver content, monitor student participation, and assess student performance online. The LMS may also provide students with the ability to use interactive features such as threaded discussions, video conferencing, and discussion forums. LMS is also called Course Management System, or CMS [1].

The transition from traditional, offline training to eLearning may sound challenging, but the outcomes are worth it. Here are some of the major features one can expect from a LMS:

Managing courses, users and roles Learning management systems may be used to create professionally structured course content. The teacher can add text, images, videos, etc. Moreover, they can create different types of users, helping control which content a student can access, track studying progress and engage student with contact tools.

Online assessment An LMS can enable instructors to create automated assessments and assignments for learners, which are accessible and submitted online.

User feedback Students exchange of feedback both with teachers and their peers is possible through LMS. Teachers may create discussion groups to allow students feedback, share their knowledge on topics and increase the interaction in course.

Synchronous and Asynchronous Learning Students can either learn asynchronously through course content such as pre-recorded videos, PDF, SCORM or they can undertake synchronous learning through mediums such as Webinars.

Learning Analytics Learning management systems will often incorporate dashboards to track students progress. They can then report on key items such as completion rates, attendance data and success likelihood.

A Learning Management System consists of two parts. The first part is an admin interface where a training manager performs the core, back-office tasks to organize their company's learning programs. This is where they create, manage and deliver courses, add learners, analyze reports, automate notifications, etc. The second part is a user interface that runs inside your browser (like Gmail or Facebook). This is what learners see when they enroll or are assigned to a course.

These platforms are used by anybody that is trying to deliver an eLearning course - this includes a lot more than only educational institutions. LMS are used in businesses of all sizes, from large multinational enterprises to small and medium businesses, businesses from a wide range of industries, such as healthcare organizations and tech startups, government organizations, traditional educational institutions (schools, universities, colleges), online and eLearning-based educational institutions and more. It's clear that a LMS can serve different training needs. For example, an expert can use an LMS to sell their courses online, a company can use it to train employees internally, while a consulting agency can train clients on a range of topics [19].

3.1.2.1 Learning Tools Interoperability

How can we connect our web application where the logic exercises will be implemented, with the course page hosted in the LMS? This is the topic we will address in this section, and we will use what is called LTI. Learning Tools Interoperability (LTI) is a specification developed by IMS Global Learning Consortium, and it establishes a standard way of integrating rich learning applications, called tools (delivered by tool providers) with platforms such as learning management systems, called tool consumers.

The basic workflow for using LTI starts when the Instructor or LMS administrator gains access to an externally-hosted learning tool. The tool's administrator provides the administrator or instructor a URL, key, and secret for the Tool [10].

- For the Instructor use case, they generally add a LTI tool into their course structure as a resource link using the LMS control panel. The instructor enters the URL, secret, and key into as meta data for the resource link. When students select the tool, the LMS uses the URL, secret, and key information to seamlessly launch the student into the remote tool in an iframe or new browser window.

- For the Administrator use case, they generally add a "virtual tool" to the LMS, entering the URL, secret and key. Once this is done, Instructors simply see the newly configured LTI tool as another tool or activity to be placed as a resource link in their course structure. The Instructors and students may not even be aware that the tool they are using is running out side of the LMS. They simply select and use the tool like any other tool that is built-into the LMS.

- In both cases, the external tool receives a Launch request that includes user identity, course information, role information, and the key and signature. The launch information is sent using an HTTP form generated in the user's browser with the LTI data elements in hidden form fields and automatically submitted to the external tool using JavaScript. The data in the HTTP form is signed using the OAuth (www.oauth.net) security standard so the external tool can be assured that the launch data was not modified between time the LMS generated and signed the data and the time that the Tool received the data.

On the topic of LTI, to completely understand what we are talking about here, we need to consider some important terminology:

- **Tool Provider:** A Tool Provider exposes interfaces to one or more tools. It is useful to think of the Tool Provider as a wrapper around a tool. In this sense, the Tool Provider represents the entire system of a Tool plus its interfaces.

- **Tool Consumer:** Instructors and students typically access Tools by activating links from within an LMS. But the LTI standard supports other scenarios, too. For example, you might want to launch a learning tool from Facebook, or your Google home page. Any system that offers access to a Tool is called a Tool Consumer.

- **Context:** Links for Tools typically appear in the context of some course. But they may appear within other types of groups. For example, links might appear within the context of a student organization, club, study group, etc. Since Tools may be launched from many different types of contexts, the LTI Specification rarely uses the term "course" and instead uses the more general term "context."

- **Resource Link:** The Tool Consumer uses Resource Link entities to generate clickable links within its user interface. Each Resource Link has a title which defines the text that should appear in the clickable link plus an optional description which should appear alongside the link.

IMS is enabling dramatically better usability and data security through LTI Advantage — a package of services that add new features to enhance the integration of any tool with any LMS. LTI Advantage includes advanced protections for exchanging sensitive student data found in core LTI 1.3. The three LTI Advantage feature services are Names and Role Provisioning Services, Deep Linking, and Assignment and Grade Services [11].

Assignment and Grade Services The Assignments and Grades service (LTI-AGS) allows LTI tools to send and manage learner grades back to the platform. When LTI-AGS is enabled and the component is set as graded in an LMS course. The LTI tool may also manage, edit, and override student's grades directly, and provide reports on the user status of the activity (pending, started, fully graded). You can enable this feature on your instance following the Assignments and Grades service setup documentation.

Deep Linking The Deep Linking service (LTI-DL) allows course creators to select and configure the content displayed to learners. This removes the need to use custom parameters and settings when setting up content, improving the ease of use and content authoring experience. For example, a tool may let course creators configure a link that will launch a specific chapter from a book rather than displaying the full book and having students manually scroll to that chapter.

Names and Role Provisioning Services The Names and Roles Provisioning service (LTI-NRPS) allows tools to list and retrieve information about the learners that have access to an LTI component. The tools can retrieve a limited amount of personal information (full name, email, username) and the membership status of all the learners enrolled in the course. A tool that uses LTI-NRPS will be aware of all users with access to that tool and can provide reports on student activity within the tool.

In most cases, Tool Consumers and Tool Providers are both web applications. They communicate with each other by making a connection using the LTI standards. In essence, connections are HTTP requests and responses.

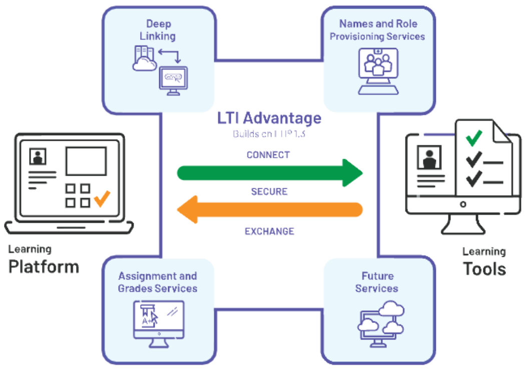


Figure 3.2: Tool consumer and tool provider graphic

3.1.2.2 Moodle

Moodle is an abbreviation of Modular Object Oriented Dynamic Learning Environment. It is an LMS that was developed in 2001 and is based on the social constructionist pedagogical principle. The characteristics the Moodle platform are user-friendliness, accessibility, and flexibility. There is also a list of students on each course allowing the lecturer to know the last time each student accessed the platform, the ability to integrate Moodle into other systems, the ability to allow synchronous and asynchronous interaction, a personal area for draft writing and journaling as well as managing personal and private information, and content that develops based on the teaching and learning needs and that can be reused [12].

As Moodle is an Open Source LMS, the cost of licensing is free. The only costs you need to take are the ones of hosting a private server. For this LMS to work, an Apache server is required to be installed along with additional PHP packages that might not be present in the host computer, as Moodle's code is written in PHP. It must be kept in mind that the owner of the Moodle directory should be the web server so that it can write to it and make changes. Moodle requires a database which can be MySQL, PostgreSQL, Microsoft SQL Server or Oracle.

An activity is a general name for a group of features in a Moodle course. Usually an activity is something that a student will do that interacts with other students and or the teacher. This is where you get the true interactivity in a Moodle course. The basic activities include assignments, chat, multiple choice questions, feedback, forum, wiki, and many more [16]. There are various other third-party modules that have been developed or enhancements to the modules above.

To integrate our web application, where the logic exercises will be located, with our Moodle course page we need to use what Moodle calls an External Tool, that essentially is a tool provider/consumer model.

External tool Many websites provide materials and interactive learning exercises different from and complementary to Moodle's own resources and activities. The external tool offers a way for teachers to link to these activities from within their Moodle course page and where available to have grades sent back into Moodle. Students only need to log in to Moodle; they do not have to log in a second time to the connecting site.

Sites which allow connection to Moodle in this way are known as LTI compliant and are called Tool providers. The "tools"are the exercises or materials which Moodle can connect to.

3.1.2.3 Open EdX

Open edX® was created by the joint efforts of Harvard University and MIT for the well-known learning platform edX. It is an open-source, learning management system (LMS) that empowers organizations worldwide to design customized and engaging online learning platforms.

Just like Moodle, OpenEdX is an open-source platform, meaning there are no license fees and the only costs you have to cover are the expenses of a private server that runs OpenEdx. Some of the most powerful features, and part of the reason as to why OpenEdX is as popular as it is, include:

- **Powerful Analytics** gives a competitive edge to Open edX over its competitors. Reports can be generated to identify problems and get timely feedback. Advanced insights help in improving the course and tracking the performance of the learners.
- **Tools and Integrations** at Open edX such as API and LTI are available. LTI tools enable communication with external software and enhance the functionality of the system.
- **Scalability** is the need of the hour in the eLearning world. Open edX platforms are designed to host a large number of learners at the same time.
- **Interactivity** enhances the entire experience of a user. With xBlocks[1], Open edX can go beyond simple interactions in an online classroom.

LTI support in the Open edX platform opens a wide range of possibilities for learning content. LTI tools are available from a wide range of organizations, from large ed-tech

[1]The XBlock specification is a component architecture designed to make it easier to create new online educational experiences. XBlock was developed by edX, which has a focus in education, but the technology can be used in web applications that need to use multiple independent components and display those components on a single web page. [17]

vendors to universities building their own integrations. In Open edX software, LTI is used in two ways: as an LTI consumer in courses to enrich the learner experience by embedding interactive problems and exercises from external sources, and as an LTI provider to allow other LMS's to embed Open edX course content.

3.1.2.4 Moodle vs OpenEdx

In this section we will compare some characteristics of both Moodle and OpenEdX, to help us make a choice on what learning management system we will use. After listing some of these comparisons, we will decide which platform to use and give an explanation as tot why that choice was made.

In terms of core features, both platforms are very similar - for our project, the two platforms are completely viable. With that said, a choice still needs to be made, and we will list the main reasons that made us choose Moodle over OpenEdx.

- While both platforms are open-source and have been around for a long time, Moodle was released around 10 years earlier then OpenEdX; consequently, Moodle's community and plugin library are very large.
- OpenEdX takes an approach more focused in the MOOC realm, focusing on the student and disconnecting the relationship between the instructor and the learner. Moodle on the other hand, is designed with a more traditional classroom environment in mind, prioritizing the instructor and learner interaction. Moodle was also not designed to have huge numbers of concurrent users, whereas OpenEdX was made to handle that much amount of traffic.
- After the initial setup of both platforms is complete, it is easier on the Moodle platform for a teacher to do what he needs to do in terms of course management. In OpenEdx, some functionalities require server access, and there is a thing called Studio, that is a dashboard for instructors to manage their course.
- Moodle has a higher level of customization than OpenEdX, allowing for a more personalized course page.

The main problem we are trying to solve is we need a platform to complement the Logic Course lectured in the Universidade Nova de Lisboa. Taking into considerations both the advantages and disadvantages of both the studied learning management systems, it was decided that the best suited platform for our needs is Moodle. Another deciding factor is the fact that the university already has a preference on using Moodle, so the transition of the project to the actual classroom would be easier.

To summarize this section, we have looked into two LMS - Moodle and OpenEdx - and reached the conclusion that the platform we are going to use is Moodle. After setting a course on Moodle, we can afterwards add the implemented exercises that are located on a different Web server with the use of LTI, making so that we have the exercises on the course page, and the instructor can use them to evaluate students and grading them.

3.2 Logic Exercises

In this section we will introduce some of the logic exercises that will be added to our online platform, and for each of them individually, we will discuss the details on the design and future implementation of the exercise. The requirements to make the exercises we want to have on our platform possible will also be given in this section, along side a better view on the importance of the tools that will be implemented discussed in 3.3.

All of the exercises presented in this section are designed in a way that when they are actually implemented and the system is online, it is very easy to add new exercises that follow a given format. The main objective of our platform is to allow for a system administrator, in this case the professor of the logic course, to add new exercises of the types that already are present in the platform. Instead of having hard-coded exercises, manually designed for a specific subject or topic of the course, the instructor can add new exercises by submitting the data for new ones in a predetermined format, and the system automatically generates the exercise with that information and adds it to the system, giving learners more exercises to solve without having to manually write the exercise on the system. For example, in an exercise where the main focus is to transform a natural language phrase into a propositional logic formula, to add a new exercise the instructor only needs to give a new phrase in natural language and associate it with a correct propositional formula, and the platform automatically generates the exercise.

3.2.1 Classical Propositional Logic

Each of the exercises will have a fixed structure, where first we discuss the details about the design of the exercise; secondly, we give detailed descriptions of the type of feedback given on each exercise; finally, we briefly discuss the details on the key interface elements of the exercises.

One of the most important aspects in this platform is the way the learner receives his feedback from the application. We need to make sure we give enough for the learner to understand where he got the exercise wrong, but we also need to make sure we don't give too much and give the answer without him understanding why. It's also important to give positive feedback and for some exercises give feedback explaining the solution, for the learners that might have gotten the right answer but aren't really sure how they got to it.

To conclude the section, we will give a brief overview on the overall system requirements for each of the exercises.

3.2.1.1 Propositional formula syntax: transforming a sentence from natural language to propositional logic

In this first exercise of Classical Propositional Logic, the main objective is to train a learner in transforming phrases in the natural language into well-formed formulas. For example, consider the phrase in natural language "Max will go fishing if it is sunny", and the propositions p (Max will go fishing) and q (It is sunny). From this example, a possible propositional formula that represents the natural language phrase is $q \rightarrow p$.

This type of exercise is important not only for the learner to practice writing the formulas that represent certain propositions, but also to help them get familiarized and comfortable with using the platform for the first time. After the learner can easily use the interface elements of this exercise, the adaptation to the other exercises and new interface elements will be easier to learn.

Design Details There are a couple of aspects we have to take into consideration while designing this exercise. To begin with, we need a way for the learner to see the phrase he needs to convert into a formula and a way for him to write that formula. The interface side of things will be covered in more detail in the coming paragraphs. Now that we have a formula that the learner wants to submit as an answer, the system needs to make sure of two things:

1. The formula submitted by the learner is well-formed.

2. If the answer submitted is a well-formed formula, then the system will test the correctness of the solution.

Before we test if the formula submitted is correct or not, we need to verify that the formula is well-formed. Because this task will be very frequently used in many of the other exercises, if not all of them, a **parser** will be implemented to make sure that all the formulas that are submitted by learners are well-formed. Further details on the parser can be found in section 3.3.

After making sure the submitted formula is well-formed, the system needs to verify if it describes the phrase in natural language of the exercise. To make this possible, when the exercise is created the instructor gives a phrase and a well-formed formula that is a possible solution to the exercise, so that it can be compared to the learner's solution. To make sure the answer is correct, we need to verify if the submitted formula is equivalent to the formula the system as associated with that particular exercise. This is done with the use of a tool that will be implemented in the future called equivalence verifier - more details on this tool can be found in 3.3.

When the system finishes the two verifications that were enumerated above, and all the appropriate feedback is given to the learner, this first exercise is over, and the process starts all over again.

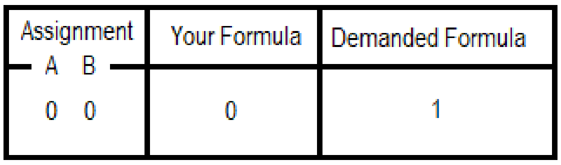
Feedback The first feedback we need to give a learner in this exercise is whether the formula he submitted is well-formed or not. During the parsing of the formula, the system will try to save information on where a possible error might have occurred whenever possible. After this process is complete, the system will inform the learner if the formula he submitted is well-formed or not and, if the learner wants, he could try and ask for a hint whenever it is possible. These hints may vary on the type of error picked up by the parser, but would mainly be informing there is missing a parentheses, or it is misplaced, or a logical connector is out of order, etc. A couple of examples of feedback for this type of scenario would be:

- There is a parentheses missing to close the expression...

In the case where the formula submitted by the leaner is a well-formed formula, but isn't equivalent to the formula associated with the current exercise being solved, the feedback given to the learner tells him that, although the formula he submitted is well-formed, it doesn't represent the natural language phrase that we want. The student can request a hint to the platform, that will try to tell him where he might have made a mistake, if possible (this type of feedback will not be possible on every single example). Examples of this type of feedback would include:

- You might have mixed up equivalence $\alpha \leftrightarrow \beta$ and implication $\alpha \rightarrow \beta$...

Regardless of whether the previous feedback is possible or not, if a learner submits a well-formed formula that doesn't represent the natural language phrase that is asked in the exercise, the system will give as feedback to the learner a possible assignment of the propositional symbols and the truth-values the correct formula should have for that assignment, in comparison with the truth-values of the formula submitted by the learner.



Assignment		Your Formula	Demanded Formula
A	B		
0	0	0	1

Figure 3.3: Feedback on the correct truth-value of an assignment

Interface Each exercise will consist in one natural language phrase being displayed on the platform, with a set of fixed propositional symbols and the corresponding propositions. An example of having fixed propositional symbols would be like this:

- **R:** It is raining.
- **S:** The sun is shining.

- **B:** A rainbow can be seen.
- **C:** It is cloudy.
- **U:** Max is carrying an umbrella.

By having the propositions fixed, we can define a set of buttons with the possible symbols that would be required to compose a well-formed formula, as shown in 3.4. Along side the answer box where the learner's formula is written, there will be a submit button for the learner to inform the system he's ready to have is answer corrected.

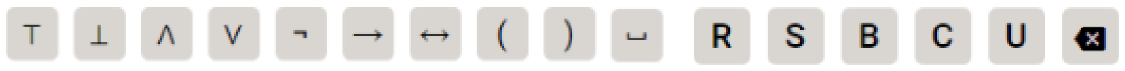


Figure 3.4: Buttons available to input a new formula

3.2.1.2 Truth tables from a propositional formula

The main objective of this exercise is for a learner to create a truth-table of a given propositional logic formula. To add different levels of complexity to this exercise, the first part of will consist in filling the blank spaces of a truth-table that already has the column names filled in, each column corresponding to a sub-formula of the given formula in the exercise. The next level of complexity, consists in the learner creating the entire truth-table, including filling in the column names corresponding to sub-formulas of the original one.

Design Details To be able to test the correctness of the solutions submitted by learners, the system will need a way to generate a correct truth-table of a given formula, to compare it to the tables submitted. For this purpose, a tool for creating truth-tables of a given formula will be implemented to help test the correctness of the solutions submitted, and will be explained in further details in 3.3.2.

The way the system checks if the solutions submitted are correct or not consists in comparing the solutions with the correct truth-tables generated by the auxiliary tool cell by cell. If both tables are an exact match, then the solution is correct; if there is a single cell different, the system gives feedback on the error.

On the second complexity level of the exercise, the learner will also need to fill in the columns of the truth-table, meaning that for the correctness of this level, not only are both tables compared cell by cell, but the system also checks if the column labels match the generated truth-table.

Feedback In the case that the truth-table submitted by the learner matches the truth-table built by the auxiliary tool, the system gives the learner positive feedback telling him everything is correct. If the answer submitted doesn't exactly match the truth-table that was generated by the system, the learner is given different levels of feedback depending on what he wants: first, the learner is told that there are N number of entries wrong, without disclosing the location of the errors. If he still needs further help, there will be an additional hint that will say exactly which columns contain an error.

Interface Each exercise will have a propositional formula that the learner will fill the corresponding truth-table, and depending on the complexity level of the exercise, the truth-table will be semi-complete. The learner will fill each cell with the associated truth-value by clicking on the cell. In the exercise where the complexity level is higher, the columns of the truth-table are filled also by the learner. Under the truth-table will be located a submit button.

3.2.1.3 Semantic equivalence: convert a formula to a normal form

The main goal of this exercise is to train a learner on how to use equivalence transformations to convert propositional formulas into equivalent formulas in either conjunctive or disjunctive normal forms.

This exercise will have two levels of complexity, where in the first level, the learner is given both an original propositional formula as well as an equivalent formula to the original, but already in either conjunctive or disjunctive normal form. Like this, the learner has a start and a finish line, and only needs to apply the correct transformations on the original formula to produce the one asked by the exercise. In the second level of complexity, the only difference is that the learner isn't given a goal formula already in a normal form, and needs to start from the original one and get to the asked normal form (conjunctive or disjunctive).

Design Details Every time a learner applies a transformation to the original formula of the exercise, the system needs to always make sure that the newly produced formula is well-formed and is equivalent to the formula the transformation was applied to. As we described in a previous exercise, this first task of making sure a formula is well-formed is a very recurring task on our platform. For that, like in the previous exercise, the task of making sure a formula is well-formed will be performed by one of the auxiliary tools that will be implemented for this dissertation - the parser (3.3.1).

After making sure the new formula is well-formed, the second thing the system needs to make sure is if that new formula is still equivalent to the one the transformation was originally applied to. For this task, the system will use an auxiliary tool designed to verify if two formulas are equivalent or not - this tool is further described in 3.3.3.

The learner repeats the process of applying these transformations to the original formula until he reaches the asked normal form in the exercise. Every time a new formula is entered in the system, it is verified that it is well-formed and equivalent to the original, and if it isn't, the system gives the leaner the appropriate feedback.

Feedback Before verifying if the answers submitted by a learner are correct, the application needs to check if the formula that was submitted is still a well-formed formula. This is verified by the parser, and depending on if the formula is well-formed or not, the system will give the learner appropriate feedback. This feedback process of checking if a formula is well-formed or not was already described in the feedback paragraph of the first exercise, so we will skip it here.

After this, the system still needs to verify that the newly submitted formula is equivalent to the original. In the case that the new formula isn't equivalent, the learner is told his transformation, although it produced a well-formed formula, it was not equivalent to the original, therefore rejecting that transformation and returning to the original.

When the learner arrives at the asked normal form, he submits the exercise to receive feedback on whether he has reached a solution or not, and the system with the use of another auxiliary tool that will be implement will verify if a given formula is in the normal form that is passed as a parameter. This tool will be further described in section 3.3.4.

If the formula submitted by the learner isn't in the normal form that was requested, he is given an example of a correct formula in that normal form so that he can try to spot his mistake.

Interface Each exercise will have a starting formula and a type of normal form that the learner is supposed to reach with transformations from the original formula. In the first complexity level, there will also be a target formula in the asked normal form, to make it easier to visualize the transformations for the learner. There will be the usual set of buttons shown in figure 3.4, so the learner can write new formulas. A submit button will also be found for a learner to receive feedback on his current written answer.

3.2.1.4 Satisfiability

In this exercise we will explore two different ways to test the satisfiability of a propositional formula, starting with the Horn-SAT Algorithm and then moving on to the resolution rule.

HORN-SAT In this part of the exercise the learner will simulate the execution of the Horn-SAT Algorithm to test if a given propositional formula is satisfiable or not. Additionally, if we can prove a formula is satisfiable, the algorithm also gives us an attribution of truth-values for the propositional symbols that satisfy the given formula.

As we have said in section 3.2, this algorithm can only be applied to Horn Formulas, which are formulas formed by a conjunction of Horn clauses. A Horn clause is a clause with at most one positive literal, called the head of the clause, and any number of negative literals, forming the body of the clause. Note that a propositional symbol α can be represented as $\top \rightarrow \alpha$ and $\neg\alpha$ can be represented as $\alpha \rightarrow \bot$.

A possible additional level of complexity would consist in having the learner start with a formula that is not an Horn formula, and he would have to apply transformations to it until he could use that formula with the algorithm.

Design Details The learner will manually click on the propositional symbols to select the next symbol to be marked by the algorithm. Every time he selects a new propositional symbol, the system will test the correctness of the answer by comparing it with the actual algorithm results that the system will run in the background.

If the learner marks an incorrect symbol, the system doesn't allow that action and immediately gives feedback on the error. He will repeat this process until all literals are marked, and if at any point the algorithm marks $\perp$, the algorithm returns false meaning the formula is not satisfiable. In the case that the $\perp$ is not marked, all the propositional symbols marked at the time are a valid attribution that satisfies the given propositional formula.

Feedback Whenever a learner marks a wrong propositional symbol, the system doesn't allow it and gives feedback on what the next symbol to be marked is. For example, if a learner tries to mark a symbol that appears in a premise of an implication, the systems tells him that symbol cannot be marked, and hints what the next one to be marked.

If the learner marks a symbol correctly, the system gives feedback that the marking was correct and gives a brief justification as to why the answer is correct. For example, we could have the following feedback:

- Step 1: The variable α has been marked due to clause α. The following propositional symbols are marked: α
- Step 2: The variable β has been marked due to clause $\alpha \rightarrow \beta$. The following propositional symbols are marked: α, β

In the case that the learner tries to submit his answer without marking all the symbols first, the system tells him there are still N symbols to be marked.

Interface This exercise will have the Horn formula on the screen, and each propositional symbol will be clickable so the learner can mark variables for the Horn-SAT Algorithm. Below the formula area, will be where the feedback is given to the student. In the end of the exercise the student has an input to say if the formula is satisfiable or not, so the system can give the final answer.

Resolution Rule In this second part of the exercise the learner will use the resolution rule to test if a given propositional formula is unsatisfiable. Unlike the Horn-SAT Algorithm, which tells us if a formula is satisfiable or not, with the resolution rule we can only take conclusions on whether the formula is not satisfiable. The learner will apply the resolution rule on a set of clauses originated from the given formula, and will try to derive an empty set. If he manages to get to this empty set, the formula is unsatisfiable; in the case that the learner can't derive an empty set, we can't say if the formula is satisfiable or not.

It is important to note that to derive the initial set of clauses from the original formula, that formula needs to be in the conjunctive normal form.

Design Details From the initial formula we need to create the set of clauses from it. After having the set formed, the learner will begin the process of applying the resolution rule on the clauses. Every time the learner tries to join two clauses with complementary literals, the system needs to make sure that it is possible to simplify a literal. For that reason, if the learner tries to join two clauses that can't be joined, the system doesn't allow it, and gives feedback on why that move isn't possible. To give an example, from the propositional formula $\alpha \wedge (\neg\alpha \vee \beta) \wedge (\neg\alpha \vee \neg\beta \vee \theta) \wedge \neg\theta$, we get the set of clauses $[\alpha]$, $[\neg\alpha, \beta]$, $[\neg\alpha, \neg\beta, \theta]$ and $[\neg\theta]$.

The learner will repeatedly join pairs of clauses until he can get to the empty set. It is possible that this process will take a couple of tries, because it can happen that the choices of clauses up to that moment may not generate an empty set, and the learner might need to try to join other complementary literals first.

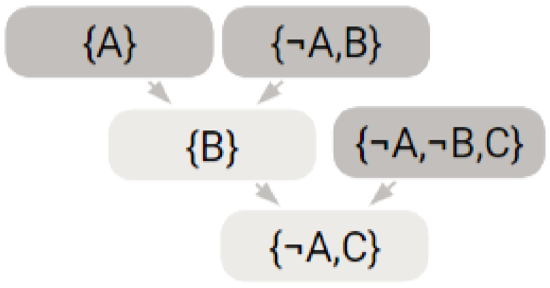


Figure 3.5: Example of applying the resolution rule and not getting an empty set

When the learner manages to derive the empty set, he can submit his answer and say the formula of the exercise is unsatisfiable.

Feedback If the learner succeeds in deriving an empty set, the system gives positive feedback saying the exercise is correct. If the learner tries to join two clauses that don't have complementary literals, the system tells him that his choice is impossible and hints on a possible pair of clauses to join. This hint is given only a certain amount of times, to avoid giving away the entire answer.

Interface The interface of this exercise will be very different from the others in the platform. For each formula, there will be a blank area with the initial set of clauses derived from the original formula. The learner will then start pairing up clauses with

Exercises	**Parser**	**Eq. Verifier**	**TT. Creator**	**NF. Verifier**
1	T	F	F	F
2	F	F	T	F
3	T	T	F	T
4	F	F	F	F

Table 3.1: Requirements table for the logic oriented tools

Exercises	**Buttons for Symbols**	**Log**	**Blank Canvas**	**Truth-tables**
1	T	F	F	F
2	F	F	F	T
3	T	T	F	F
4	F	F	T	F

Table 3.2: Requirements table for the interface elements

complementary literals, by selecting both clauses that contain those literals and forming a new clause on the exercise area. After the process of deriving the empty clause is over, the learner can submit his answer on if the formula is unsatisfiable.

3.2.2 Requirements

In this subsection we present the requirements of each exercise in the following tables. First we begin with the logic oriented tools that will be implemented, showed in table 3.1. The requirements for the interface elements are shown in table 3.2.

3.3 Logic oriented tools to be implemented

In this section we are going to discuss the logic oriented tools that will be created to make the implementation of the exercises mentioned in section 3.2 possible. Starting in subsection 3.3.1, we talk about the importance of implementing a parser, that will be used to test if the formulas in our system are well-formed or not. Next we discuss a tool used for creating truth-tables of a given well-formed formula in 3.3.2. To conclude this section, we discuss two tools used for verifying certain conditions - we have one to see if two formulas are equivalent, and another one to see if a given formula is in the normal form that was asked - both can be found, respectively, in 3.3.3 and 3.3.4.

3.3.1 Parser

In this subsection we will discuss the details of the parser that will be implemented for this application. A parser is a software component that takes input data and builds a data structure – often some kind of parse tree, abstract syntax tree or other hierarchical structure, giving a structural representation of the input while checking for correct syntax.

After looking at the exercises showed in 3.2, there is a clear need to verify if a given formula is well-written or not, for the design and implementation of some exercises to be possible. For example, on the first exercise where the goal is to transform a natural language phrase into a propositional formula, there needs to be a way to verify if the formulas submitted by learners are well-written or not.

There are solutions to this problem already found in third-party libraries, such as ANTLR[2], but most of them would only be able to tell us if a given formula is well-written or not, in the form of a Boolean variable. This wouldn't be a problem if we didn't want to give a custom feedback on the parsing of the formulas - for example, if a formula was missing a parentheses, the feedback given to the learner would indicate the error was a missing parentheses. What this means is that during the parsing process, we will need to save information on the error so that we can give a more accurate feedback to the learner.

The first thing that we need to do is create a grammar of the language of we want our parser to read - in this case, Classical Propositional Logic - so that it can use the rules defined in the grammar to make sure a formula is well-written according to the Propositional Logic rules. Further details on the practical implementation of the parser will be given when the implementation phase of the dissertation begins.

3.3.2 Truth-Table Creator

In this subsection we will discuss the details of the tool that will be implemented to create automatically the truth-table of a given well-formed formula. There are a couple of reasons that would justify the creation of this tool, starting with the first - in the exercise where the learner fills out a truth-table manually of a given formula, we need to have a correct version of that truth-table built automatically to test the correctness of the solution. Secondly, and arguably most importantly, this tool will be very useful with the implementation of another tool in this section - the equivalence verifier. More details on this topic will be given in 3.3.3.

For this functionality, we could make use of a third-party library as we simply need a tool that gives us a data structure with the truth-table values. The only concern in using a third-party tool would be to manage and convert inputs and replies to match our systems. But if that work proves to be less than actually creating the tool itself, it would be a valid option to consider using a third-party tool.

Further details on the practical implementation of the truth-table creator will be given when the implementation phase of the dissertation begins. There we will decide on whether it is better to use a third-party tool or to create our own.

[2]**What is ANTLR?:** ANTLR (ANother Tool for Language Recognition) is a powerful parser generator for reading, processing, executing, or translating structured text or binary files. It's widely used to build languages, tools, and frameworks. From a grammar, ANTLR generates a parser that can build and walk parse trees. [2]

3.3.3 Equivalence Verifier

In this subsection we will discuss the tool that will be implemented to verify if two given well-written formulas are equivalent or not. As we mentioned in 2.2, we can prove two propositional formulas are equivalent by comparing their truth-values. If it is the case that two formulas have the same truth-tables, then it means they both are equivalent; if the truth-tables are different, we can't say they are equivalent.

The implementation of a tool like this is very simple and will not require the need for a third-party library - given two well-written formulas, the application generates a truth-table for each of them and compares the results cell by cell, making sure all elements are equal; if it is the case, then both formulas are equivalent, if not then both formulas are not equivalent. The only challenge lies in the creation of a truth-table given a formula. Fortunately, this tool is already going to be implemented and is better described in 3.3.2.

3.3.4 Normal Form Verifier

In this subsection we will discuss the tool that will be implemented to verify if a given formula is in the normal form that we need. For example, the tool receives a propositional formula and a type of normal form as inputs and returns a Boolean variable telling us if the formula is in the normal form that was given as a parameter.

This tool will prove to be very useful in the exercise where the learner needs to apply equivalence transformations on a given formula until he manages to convert it to the request normal form. When the learner submits his answer, the type of normal form asked in the exercise and the formula he wrote are sent through the tool which tests the correctness of the learner's answer.

There are some third-party tools that can verify if a given formula is in the asked type of normal form or not, and it would be an option to consider in the implementation phase of the dissertation. The only problem with using an outside library is we would need to manage and convert inputs and replies to match with our system, but if that work proves to be less time consuming that implementing the tool itself, than it would be a good solution. Further details on the practical implementation of the normal form verifier will be given when the implementation phase of the dissertation begins. There we will decide on whether it is more benefitial to use a third-party tool or to implement our own tool.

4 Proposed Solution

Having introduced the problem and over viewed related work, in this final chapter we discuss the proposed solution. First, we briefly talk about the platform. Afterwards, we present the work plan for this dissertation. Finally, we conclude by unveiling the corresponding task schedule.

4.1 Online Platform for Logic

We now revisit the problem stated in the Introduction and discuss the details of our proposed solution - a online platform for learning Logic. We're going to revisit the systems architecture and the main tools that will need to be implemented along side with the integration of the exercises created onto a Moodle course. This process will be accomplished with the use of Learning Tools Interoperability specifications.

Online learning platforms play an important role in modern education. However, they may not be sufficiently represented in educational institutions as much as they would like. This includes the University of Nova Lisbon, which manifested it's interest on the development of an online platform to aid the classes of the Logic courses on the university, in the form of this dissertation. The instructors wanted a tool that learners could use to practice common logic exercises, as well as a way to automatically grade students through a platform such as Moodle, but that was more complex than the usual quizzes and tests that one can normally find there.

As was mentioned in section 3.1 in greater details, our system will have a three tiered architecture. Each tier will be implemented using the best technologies available, and the ones currently selected can be subject to change later on if during the implementation phase of the dissertation we realize there is a better tool available for the job. After implementing the platform which contains the Logic exercises that were discussed in section 3.2, we will integrate those exercises onto a Moodle course and establish a connection between Moodle and our platform, allowing for the exchange of data through what is called LTI. This allows for Moodle to use exercises from this outside tool to be used to grade students that use the Moodle platform.

We will also implement a set of logic orient tools that will be used for the future implementation of the exercises on the platform. Each tool can be implemented in a yet to be determined programming language, that will be decided upon the implementation process. We can also use third-party tools available from other developers if it proves to be more efficient and reliable than implementing our own version. For some tools we can't simply use an outside tool, because there are certain very specific functionalities that we will need to implement for ourselves.

Before starting the implementation of the platform, we will verify the system requirements extensively and we will add additional Classical Propositional Logic exercises as well. There are also plans to add the subject of First-Order Logic and consequently the associated exercises.

To formalize everything, the main goal of this dissertation is the development of an online platform to solve Logic exercises, and integrate those exercises on a LMS to allow for instructors to conduct automatically graded assignments and quizzes to evaluate students. It is important that each task is very carefully thought through and is implemented correctly. For that there will be a lot of testing to make sure everything is functional and comprehensible. We expect some challenges and difficulties in the implementation and design of some of the exercises and tools required to make the platform functional, as well as the integration of those exercises on the Moodle course.

4.2 Work Plan

In order to achieve the objectives outlined in section 1.2 and address the questions raised in the previous section, we split the dissertation into three sequential tasks plus a forth continuous one:

1. **New Exercise Design and First-Order Logic**

   The aim of this task is to add the subject of First-Order Logic to the dissertation. A background on the important topics will be given, and new exercises will be designed and implemented in the next phase of this dissertation. Along side with the new First-Order Logic exercises, there will be added some new exercises of Classical Propositional Logic too.

2. **Online Platform Development**

   The goal of this second task is to start developing the components of the system, and initiate the practical implementation of the exercises of the platform. This task will be divided into the implementation of three separate components: the presentation layer, the application layer and the data layer, each implemented with the best tools available.

3. **LMS Integration**

   This task will consist in establishing a course on Moodle and integrating in it the exercises that we will implemented in the online platform. This will also include a comprehensive guide on how this integration is done.

4. **System Testing**

   In this forth task we will test our system intensively to find out any major errors that need to be fixed before the system can be delivered. This will include software bug testing, and different levels of testing including outsiders from the project.

5. **Dissertation writing**

   The writing of the dissertation will be a continuous process that will started simultaneously with the development of the platform and the process of adding new exercises and Logic concepts to the dissertation. A fundamental first step will be to lay out the document structure to have a clear notion of the work that needs to be done. Another important aspect is to write important details during implementation so that the information can be clear in the final document. The chapters 2 and 3 will also be reviewed and new information will be added.

4.3 Task Schedule

The proposed schedule for the work plan and respective tasks is shown in Figure 4.1.

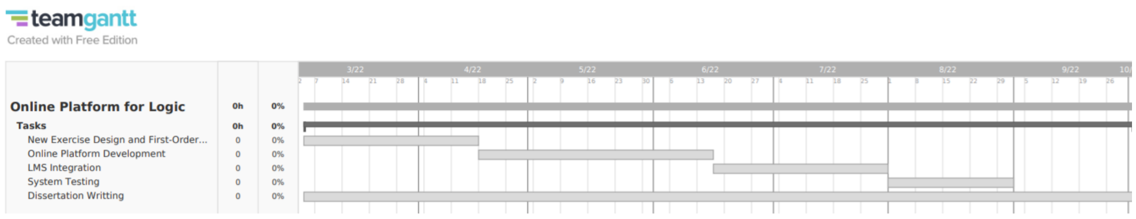


Figure 4.1: Gantt chart illustrating the proposed work plan for the dissertation.

Bibliography

[1] N. A. Alias and A. M. Zainuddin. "Innovation for better teaching and learning: Adopting the learning management system". In: *Malaysian online journal of instructional technology* 2.2 (2005), pp. 27–40 (cit. on p. 18).

[2] *ANTLR ANother Tool for Language Recognition*. https://www.antlr.org/. Accessed: 2022-02-07 (cit. on p. 33).

[3] D. Barker-Plummer, J. Barwise, and J. Etchemendy. *Language, Proof, and Logic: Second Edition*. 2nd. Center for the Study of Language and Information/SRI, 2011. isbn: 1575866323 (cit. on p. 7).

[4] Y. Chen and M. Zhang. "MOOC Student Dropout: Pattern and Prevention". In: *Proceedings of the ACM Turing 50th Celebration Conference - China*. ACM TUR-C '17. Shanghai, China: Association for Computing Machinery, 2017. isbn: 9781450348737. doi: 10.1145/3063955.3063959. url: https://doi.org/10.1145/3063955.3063959 (cit. on p. 5).

[5] *Coursera*. https://www.coursera.org/. Accessed: 2022-02-12 (cit. on p. 6).

[6] *Coursera Coursera files for IPO amid online learning boom*. https://www.cnbc.com/2021/03/05/coursera-files-for-ipo-amid-online-learning-boom-.html. Accessed: 2022-02-12 (cit. on p. 6).

[7] *EdX*. https://www.edx.org/. Accessed: 2022-02-12 (cit. on p. 5).

[8] W. Feng, J. Tang, and T. X. Liu. "Understanding Dropouts in MOOCs". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 33.01 (July 2019), pp. 517–524. doi: 10.1609/aaai.v33i01.3301517. url: https://ojs.aaai.org/index.php/AAAI/article/view/3825 (cit. on p. 5).

[9] *IBM 3 tier architecture*. https://www.ibm.com/cloud/learn/three-tier-architecture. Accessed: 2022-02-01 (cit. on p. 16).

[10] *IMS Activities*. https://www.imsglobal.org/basic-overview-how-lti-works. Accessed: 2022-02-02 (cit. on p. 19).

[11] *IMS LTI Advantage.* https://www.imsglobal.org/lti-advantage-overview. Accessed: 2022-02-02 (cit. on p. 20).

[12] N. N. M. Kasim and F. Khalid. "Choosing the right learning management system (LMS) for the higher education institution context: A systematic review." In: *International Journal of Emerging Technologies in Learning* 11.6 (2016) (cit. on p. 21).

[13] Z.-Y. Liu, N. Lomovtseva, and E. Korobeynikova. "Online Learning Platforms: Reconstructing Modern Higher Education". In: *International Journal of Emerging Technologies in Learning (iJET)* 15.13 (July 2020), pp. 4–21. ISSN: 1863-0383. URL: https://www.learntechlib.org/p/217605 (cit. on p. 1).

[14] Z.-Y. Liu, N. Lomovtseva, and E. Korobeynikova. "Online Learning Platforms: Reconstructing Modern Higher Education". In: *International Journal of Emerging Technologies in Learning (iJET)* 15.13 (July 2020), pp. 4–21. DOI: 10.3991/ijet.v15i13.14645. URL: https://online-journals.org/index.php/i-jet/article/view/14645 (cit. on p. 5).

[15] Y. Luo, G. Zhou, and J. Li. "Comparing the Chinese University MOOC Platform to the Three Major MOOC Players". In: *Proceedings of the 2nd International Conference on Computer Science and Application Engineering*. CSAE '18. Hohhot, China: Association for Computing Machinery, 2018. ISBN: 9781450365123. DOI: 10.1145/3207677.3277920. URL: https://doi.org/10.1145/3207677.3277920 (cit. on p. 6).

[16] *Moodle Basic Overview of how LTI ® Works.* https://docs.moodle.org/311/en/Activities. Accessed: 2022-02-02 (cit. on p. 21).

[17] *OpenEdX OpenEdX Documentation.* https://openedx.org/community/documentation/. Accessed: 2022-02-03 (cit. on p. 22).

[18] J. Sheard et al. "MOOCs and Their Impact on Academics". In: *Proceedings of the 14th Koli Calling International Conference on Computing Education Research*. Koli Calling '14. Koli, Finland: Association for Computing Machinery, 2014, pp. 137–145. ISBN: 9781450330657. DOI: 10.1145/2674683.2674700. URL: https://doi.org/10.1145/2674683.2674700 (cit. on p. 4).

[19] *TalentLMS Discover how — and why — companies use Learning Management Systems to train their teams.* https://www.talentlms.com/what-is-an-lms. Accessed: 2022-02-03 (cit. on p. 19).

[20] *TU Dortmund University Iltis: Learning Logic in the Web.* https://iltis.cs.tu-dortmund.de/Logic-WiSe2020-external/. Accessed: 2022-02-12 (cit. on p. 6).

[21] *UNCTAD How COVID-19 triggered the digital and e-commerce turning point.* https://unctad.org/news/how-covid-19-triggered-digital-and-e-commerce-turning-point. Accessed: 2022-02-12 (cit. on p. 1).