Tools for Learning: Computation and Logic

Karnaugh mAPP

Arkadiusz Mikolajczak

4th Year Project Report
Software Engineering with Management
School of Informatics
University of Edinburgh

2018

Abstract

Informatics 1 - Computation and Logic is the first-year course at the University of Edinburgh, mainly aimed at Computer Science students. Since it is one of the first courses students encounter at university, which introduces a lot of abstract concepts, they tend to find it complicated. To facilitate learning of the Karnaugh map model, which is one of many topics in the course, an interactive web-based tool, `Karnaugh mAPP`, was created. The following report first proposes useful features for any learning tool based on different types of learning and the gamification techniques. Then it describes `Karnaugh mAPP`'s features, including all its sections and types of exercises it offers. Subsequently, technical details of the application are presented, such as the development tools used and the implementation of crucial classes and methods. Finally, the project's output is evaluated using the information systems success model. The evaluation consisted mostly of functional testing, which involved unit tests, and usability testing conducted via an on-line questionnaire.

Acknowledgement

Firstly, I wish to thank my project supervisor, Professor Michael Fourman, for his support and feedback given at meetings held always in a pleasant atmosphere.

I would also like to thank my parents for their continuous support and all my friends who made this year much easier.

Table of Contents

Chapter 1

Introduction

1.1 Motivation

Informatics 1 - Computation and Logic (INF1-CL) is a preliminary course for all Computer Science students enrolled at the University of Edinburgh. It introduces concepts of computation and specification using propositional logic and finite-state systems.

Since higher education institutions put more emphasis on individual learning than high schools, first year students need appropriate tools to efficiently absorb the course material on their own. INF1-CL, as the primary first-year course, should provide students with such an instrument to enable them to practise and improve their skills and understanding of concepts discussed during lectures. It may as well be used as preparation for tutorials and the exam.

Computation and Logic course differs from other Informatics courses in a way that it provides mainly theoretical background used in further studies. Students cannot often think of practical usage of learned concepts, which hinders understanding of the material. In such case, tools which require the student's interaction are desired.

1.2 Chosen Topic

The course consists of a broad range of topics on logic and how it may be used in computational problems. It introduces propositional logic, which involves basic concepts, such as sets, Venn diagrams and Boolean algebra, and then more advanced operations on them, e.g. satisfiability problem and rules of inference. Further topics focus on finite state machines. The latter has already been a subject of an honours project in previous years, resulting in a decent interactive tool letting students get familiar with deterministic and non-deterministic automata, thus, the scope limits to propositional logic.

Students often practise the learned material solving past exam papers or tutorial questions. However, the Karnaugh map was introduced in INF1-CL only in the academic

year 2015/16. As it is a fairly new concept in the course, there are not many learning aids for self-studying. The interactive tool would allow students to go through the topic on their own, getting instant feedback.

The Karnaugh map is a powerful method to simplify Boolean algebra expressions. Not only does it have many applications in software design, but also in electronic circuit construction. Therefore, it is useful for Electronics and Electrical Engineering students too. Since the Karnaugh map is such a useful tool used in many engineering disciplines, it is worth making it more familiar to students, who may be overwhelmed with a new material introduced in the course.

1.3 Karnaugh Map

The Karnaugh map is a tool used to simplify complex Boolean algebra expressions. Essentially, it is a diagram that displays the truth table in an unconventional arrangement so as to make the minimised form apparent. The method was proposed by E. W. Veitch [40] and modified slightly by M. Karnaugh [24].

A	B	C	D	Cell
0	0	0	0	*a*
0	0	0	1	*b*
0	0	1	0	*c*
0	0	1	1	*d*
0	1	0	0	*e*
0	1	0	1	*f*
0	1	1	0	*g*
0	1	1	1	*h*
1	0	0	0	*i*
1	0	0	1	*j*
1	0	1	0	*k*
1	0	1	1	*l*
1	1	0	0	*m*
1	1	0	1	*n*
1	1	1	0	*o*
1	1	1	1	*p*

		AB			
		00	01	11	10
CD	00	*a*	*e*	*m*	*i*
	01	*b*	*f*	*n*	*j*
	11	*d*	*h*	*o*	*l*
	10	*c*	*g*	*p*	*k*

Figure 1.1: Truth table and Karnaugh map with labelled corresponding cells and marked possible groups.

Karnaugh maps are usually used for expressions involving two, three or four variables. Figure 1.1 shows the correspondence between the Karnaugh map and the truth table for a general case of a four-variable problem. Each cell of the diagram represents different conjunction of all variables. For example, in Figure 1.1, the square a (0000) represents the term $\neg A \neg B \neg C \neg D$, and the square n (1101) represents $AB \neg CD$.

Row and column parameters are ordered according to the Gray code, also known as the reflected binary code [25]. This means they create a sequence in which two successive binary values differ by only one bit. This makes two adjacent cells differ by the value of only one literal. When the diagram describes three or four variables, columns should be labelled as $00, 01, 11, 01$ instead of $00, 01, 10, 11$ so that columns two and three differ only by the value of A. Such a layout is the core of the Karnaugh map. Terms that differ in precisely one variable, i.e. are represented by adjacent cells, can be combined. Groups, however, must contain 2^n cells. For example, $ABCD$ and $ABC\neg D$ can be combined since $ABCD \vee ABCD\neg D = ABC$. It also needs to be noted that opposite edges are treated as adjacent. The Karnaugh map in Figure 1.1 shows some possible groups of terms.

1.3.1 Grouping States

In the process of minimising the expression using the Karnaugh map, tho following rules for grouping terms must be obeyed:

1. Groups must consist of only the same values (only *true*'s or only *false*'s)
2. Groups must be horizontal or vertical, but never diagonal
3. Groups must consist of 2^n cells (1,2,4,8 etc.)
4. Groups should be as large as possible
5. Each *true* (or *false*) must be in at least one group
6. Groups may overlap
7. Groups may wrap around the table (since opposite edges are, in fact, adjacent)
8. There should be as few groups as possible

1.3.2 Producing CNF and DNF

Each group represents the conjunction of some literals. These, in turn, can be joint in larger expressions in disjunctive or conjunctive normal form. Disjunctive normal form (sum of products) is the disjunction (sequence of ORs) of conjunctions (sequence of ANDs) of one or more literals. Similarly, conjunctive normal form (product of sums), also known as clausal normal form, is the conjunction of disjunctions of one or more literals, otherwise known as clauses or constraints. The former may be obtained by grouping all the states for which the initial expression evaluates to true and simply joining the groups with ORs. The latter requires grouping the states *excluded* by the initial expression and producing their DNF, then negating the whole expression, by applying DeMorgan's laws, representing the initial statement in CNF.

1.4 Existing Tools

There exist a few stand-alone desktop and mobile applications providing the Karnaugh map instant solver, which automatically group selected states and represent them in a desired minimal form. Installing any software often discourages the user from using it as it requires extra effort and space, hence, these tools are not looked into. Regarding web-based applications, there are many websites that describe the usage of the Karnaugh map and a few publicly available instant solvers. Analysis of four of them, which are the highest in the list of Google search results for query ”Karnaugh Map Solver”, shows they all possess at least one of the following highly undesirable features:

- User can only interact with a tool by selecting states on the Karnaugh map for which the final expression is supposed to evaluate to true, thus, they cannot provide a custom initial expression they wish to minimise or transform to DNF/CNF. [1, 2, 7, 17]
- Solver outputs the expression only in DNF. To produce CNF, the user would have to select states excluded by the final expression, produce DNF and then negate it themselves. [2, 7, 17]
- Solver groups states incorrectly resulting in more groups than required, which is against the main purpose of the Karnaugh map. [2, 7]
- Solver outputs only one final expression, even when multiple solutions are possible. [1, 2, 7, 17]
- Solver produces instant solution without any explanations of the steps taken. [1, 7, 17]
- Website provides information on the Karnaugh map or the instant solver, yet does not provide the user with any exercises to practise. [1, 7, 17]
- Graphical user interface is unpleasant, worsening user experience. [1, 7]
- Tool does not follow the responsive web design making it hard to use on the mobile devices. [7]

This project tries to address all these issues creating a tool which would teach the user how to use the Karnaugh map in an interactive and enjoyable way.

1.5 Summary of Work Performed

The project consisted of three main highly-overlapping stages:

1. Reviewing literature on how people learn
2. Developing a web-based application
3. Evaluating the application

A description of the whole process in this report is divided into chapters as follows:

Chapter 2: Learning Aspect
Review of the literature on how people learn and the techniques to facilitate this process through the interactive tool.

Chapter 3: Application Features
Walk through how the application is divided and the features it provides to the user and the description of all types of exercises.

Chapter 4: Implementation
Technical aspect of the application; programming tools used in the development and design of the application.

Chapter 5: Evaluation
Assessment of the application against the successful information system criteria and the initial goals, summary of the feedback gathered via user testing.

Chapter 6: Further Work
Discussion on what could be improved and what other features could be implemented had the time allowed.

Chapter 2

Learning Aspect

2.1 Psychology behind Learning

The goal of the project was to build a learning tool for Computation an Logic course. To create an efficient application which will increase students' understanding of the selected concept of the course, one needs to consider how humans learn. Learning is *the relatively permanent change in knowledge or behaviour that is the result of experience* [8]. Although the notion of learning goes far beyond acquiring knowledge in classes or studying for the exam, the tool's objective is to teach students (change their knowledge) how to use the Karnaugh map through a set of exercises (experience). Therefore, all principles of learning still apply.

One can distinguish three main types of learning. Its fundamental aspect is the process of conditioning, i.e. the ability to connect stimuli (changes occurring in the environments) with responses (behaviours or other actions). Other types include learning through insight as well as observation [8].

2.1.1 Learning by Association: Conditioning

2.1.1.1 Classical Conditioning

At the beginning of the 20th century, Russian physiologist Ivan Pavlov accidentally noticed an interesting phenomenon in dogs he was studying. Namely, they began to salivate once the person who normally fed them entered the room even though they had not received any food yet. Pavlov realised the dogs knew they were about to get fed in advance, i.e. they associated the appearance of the technicians with the food that followed their arrival. Pavlov decided to look into this process in more detail by carrying out a series of experiments. This led to identifying a learning process called classical conditioning - *learning that occurs when a neutral stimulus becomes associated with a stimulus that naturally produces a behaviour* [8]. The classical conditioning concerns only natural, biological responses and does not involve learning anything new. Thus, it is not effective in the process of studying the material of the course.

2.1.1.2 Operant Conditioning

Operant conditioning, on the contrary to the classical conditioning, is *learning that occurs based on the consequences of behaviour*. The pioneer of this field, psychologist Edward L. Thorndike, after a number of experiments, came up with his *law of effect* which states that *responses that are accompanied or closely followed by the satisfaction in a particular situation are more likely to recur in a similar situation, whereas responses that are accompanied or closely followed by discomfort are less likely to occur again in the situation* [38]. This idea was studied further by the psychologists B. F. Skinner. He described the operant learning using terms such as reinforcer and punisher resulting in the reinforcement theory. There are four learning (teaching) strategies emerging from this theory (shown with examples drawn from games) [32]:

Positive Reinforcement
Desired behaviour is strengthened by bestowing a positive consequence
E.g. User gains points for a correct answer

Negative Reinforcement
Desired behaviour is strengthened by reducing a negative consequence
E.g. A correct answer removes a negative score multiplier (given a multiplier was first activated)

Positive Punishment
Undesired behaviour is weakened by bestowing a negative consequence
E.g. A series of incorrect answers activates a negative score multiplier

Negative Punishment
Undesired behaviour is weakened by reducing a positive consequence
E.g. User does not gain any points for an incorrect answer

The important relation between the reinforcement and punishment is that the use of the former is virtually always more effective than the latter. The subject of the reinforcement tends to feel better creating a positive bond between them and the reinforcement provider. On the other hand, effects of punishment tend to diminish once the punishment provider quits.

There are also various types of reinforcement regarding the frequency of a desired behaviour being reinforced. In general, one can distinguish between continuous and intermittent schedules of reinforcement. The former, which reinforces a desired behaviour every time it occurs, takes less time to impose any behaviour, yet is more prone to extinction. This happens because the subject may be used to being rewarded the reinforcement so much it gives up a desired behaviour immediately the reinforcement does not appear. The latter can be divided further into four categories, such as fixed ratio, variable ratio, fixed interval and variable interval. Nevertheless, they are utterly irrelevant for this project. Detailed information on this topic may be found in the book *Introduction to Psychology* published by the University of Minnesota [8].

2.1.2 Insight

Not all kinds of learning may be explained using only conditioning. Sometimes, when tackling a problem, one comes up with a solution out of nowhere. This is known as *insight - the sudden understanding of a solution to a problem*. It is hard to formally explain the process of generating a solution. However, there is often a correlation between such learning and motivation. Namely, one can tend not to demonstrate a right solution (learning is not reinforced) until there is motivation to do so [8]. Learning by insight is an interesting aspect of animals' learning process, yet it is not a useful concept in a learning tool. The aim of such a tool is to guide the user to come up with a correct solution, not to give them an exercise unrelated to the material for which they would not even know where to seek any hints.

2.1.3 Observation

The last type of learning is observational learning, also known as modelling. It is *learning by observing the behaviour of others*. It is the common, if not the most common, way of studying the material of the course at school. Teachers provide a lot of examples concerning a given module so that students know how to tackle similar problems themselves. As part of the preparation for the exams, students often go through the questions in past exam papers since they want to learn how to approach the new, yet similar, exercises in future.

2.2 Gamification

Knowing how people learn is not sufficient to create an effective learning tool. The tool becomes useless if potential users do not actually use it. This raises a challenge how to make it more attractive to students so that they are motivated to use it.

As technology has been developing, so have information systems, which now can be found nearly in every aspect of people's lives. Nevertheless, many of them are used only out of necessity, particularly in an educational context. This makes the user easily demotivated and unwilling to open or continue to use the application. Many industries tackled this issue and decided to seek the solution in the increasing popularity of digital games which according to studies provided an evidence for their motivational appeal [23]. This led to a term called *gamification - the use of game design elements in non-game contexts* [14].

In order to introduce a gamification in the system, one must first define what a game is. A few definitions were combined to come up with four fundamental features a game should possess [28]:

- Goal - to give a purpose for playing
- Rules - to impose limitations on the user

- Feedback system - to guide the user how to achieve the goal
- Free will - to allow the user opt in or out of the game

2.2.1 Underlying Motivation Theory

The way the gamification affects people's motivation can be described using the self-determination theory (SDT) [37]. It identifies two kinds of motivation:

- Extrinsic motivation - determined by external sources, such as pressure or rewards, e.g. money, trophies or positive feedback
- Intrinsic motivation - determined by internal needs, such as self-realisation

Although both types enhance the individual's performance, the latter has a bigger impact on learning outcomes and the effort people put into a given task. Therefore, gamified systems should contain elements which would stimulate the individual's intrinsic motivation [29]. These elements include competence, autonomy and relatedness. Competence denotes the control of the outcome of the individual's actions and their impact on the environments. Autonomy makes the individual's actions deemed to be the outcome of their own decisions. Relatedness is the need of being part of the community. It should be noted that the fulfilment of only one of these needs may not be sufficient to increase intrinsic motivation. For example, even though one completes a task with a positive outcome and is praised for it, intrinsic motivation may decrease if the individual views his or her behaviour as controlled [12]. This can be improved by introducing elements of customisation [33] .

2.2.2 Gamification Design Principles

In order to increase the user's motivation and engagement, one can identify particular components of the game, also known as game mechanics, which enhance the gamified system. The most common ones are points, levels and leaderboards, categorised as goal metrics since they all are used to keep track of the user's progress and give the instant feedback [41]. Multiple studies showed these mechanics were satisfying people's intrinsic need for competence, provided they are presented as uncontrolled, e.g. the user may adjust settings [36, 21].

Points, levels and leaderboards are only some of the examples of the game mechanics. In fact, there are many more of them, each supporting specific game design, chosen to satisfy some human desire. Table 2.1 shows selected game design principles with examples of game mechanics assisting that principle [27, 15]. It also specifies what human needs are associated with these principles, which, when satisfied, increase the intrinsic motivation.

One gamified system does not need to follow all of these design principles and comprise all the mechanics. These should be selected appropriately to the requirements and the size of the software.

Design principle	Game mechanics	Satisfied need
Goal		Competence, achievement
Challenges and quests (intermediate goals)	Unlocking content, boss fights	Competence, achievement
Customisation	Difficulty tailored to the user's skill level, avatar customisation	Autonomy, self-expression
Progress	Progress bar, points, levels, virtual goods	Competence, completion, rewards
Feedback	Points, alerts, banners, jingle	Competence
Competition and cooperation	Leaderboards, levels, forum, badges, avatars	Competence, autonomy, relatedness, status, achievement, competitions, altruism
Visible status (reputation, social credibility, recognition)	Leaderboards, levels, badges, avatars	Competence, autonomy, relatedness, status, achievement, competitions, altruism
Accrual grading	Points	Competence, rewards, achievement
Freedom of choice	Multiple routes to achieve the goal	Autonomy
Freedom to fail	Allow multiple attempts	Competence, achievement, completion
Time restriction	Countdown clock	Competence, achievement
Storytelling, roles	Avatars, virtual identity	Relatedness, self-expression

Table 2.1: Gamfication design principles.

2.3 Desired Application Features

Having gone through the psychological background of how humans learn and how to increase their motivation by gamification, one can apply this knowledge to create an effective tool for learning of the Karnaugh map.

First of all, the tool needs to provide **step-by-step examples** of how Karnaugh map should be used to simplify Boolean algebra expressions or convert them into DNF or CNF. There could be only one or two complete examples so that the user understands the basic concept and has the reference to go to when encountered a problem.

The learning tool might also provide the user with a possibility to resolve any Karnaugh map related problem instantly, by implementing an instant solver. This would give an answer straight away, hence, would not be effective as a learning tool. However, the user could find a solution to any problem they might find in a book and struggle with, and the solution may be treated as a **model** that guides the user in future problems.

The application should let the user **practise** various problems related to the Karnaugh map **in an uncontrolled manner**. The user could attempt a question as many times as they wish, giving them the **freedom to fail**, and each time they would be given **instant detailed feedback**. If the provided answer was incorrect, the feedback should be informative enough for the user to realise where the mistake was made, thus, it would guide them to the right answer. Following the idea of **learning by observation**, this kind of practice as well as step-by-step examples and instantly solved problems would improve the user's skills at dealing with similar problems in future.

Step-by-step examples and practice exercises with instant feedback are effective provided the user is willing to complete them. Encouraging students to do so might be challenging, therefore, gamification may be introduced to increase their motivation. Along with the practice section there could be a so-called **quiz** section with questions similar to the practice ones, however, only with the **right/wrong feedback**, i.e. the user would be informed only whether their answer was correct or incorrect. For each correct answer, the user would be awarded **points**, which is the essence of the quiz. Points work as **positive reinforcement** since they are a reward for a desired behaviour (knowledge of the Karnaugh map). The **goal** of the quiz would be to gather all possible points, i.e. answer all questions correctly. The total score could be continuously presented as the **progress bar**, which would motivate the user to **complete** the entire quiz. Some of the questions could be **time-limited**, yet this might put too much pressure on the user, which, consequently, would make them frustrated.

The quiz may also implicitly introduce the concept of **levels**. Levels could consist of exercises of the same type or be the mixture of various questions. When progressing, new exercises would be unlocked, resembling the **quests**. Having completed each level, the user could be awarded a **badge** or some other surprising content, such as trivia, the course-related joke, or a mini-game.

The application could also collect scores of all users and display them in a **leaderboard**, giving a chance to compare progress across all users, thus, introducing a feeling

of **competition**. Nevertheless, the application of this size may not be worth implementing the competition mode where users could see the results of others.

Chapter 3

Application Features

The name of the application is **`Karnaugh mAPP`** to indicate it is an application concerning the Karnaugh map. Its current live version can be found at:

`https://karnaugh-mapp.herokuapp.com`

The application is licensed under the Creative Commons Attribution-NonCommercial 4.0 International License [11].

3.1 Main Sections

The whole application consists of seven main sections. The user can navigate to any of them from any place in the application by using the navigation bar at the top.

3.1.1 Home

Home is the welcome page the user sees, having opened the website. It provides the general information on the tool such as the features the user can find here along with the links to them.

3.1.2 Tutorial

Tutorial page is designated for people who are new to the concept of the Karnaugh map or are still unsure how to use it to convert the Boolean algebra expression to DNF/CNF. It first provides the general information on what the Karnaugh map is, illustrating its correspondence with the truth table. Afterwards, the steps how to properly group the states are listed. Each step is supported with a visual example of the correct action presented against the incorrect one. A fragment of the screenshot of this page may be seen in Figure 3.1.

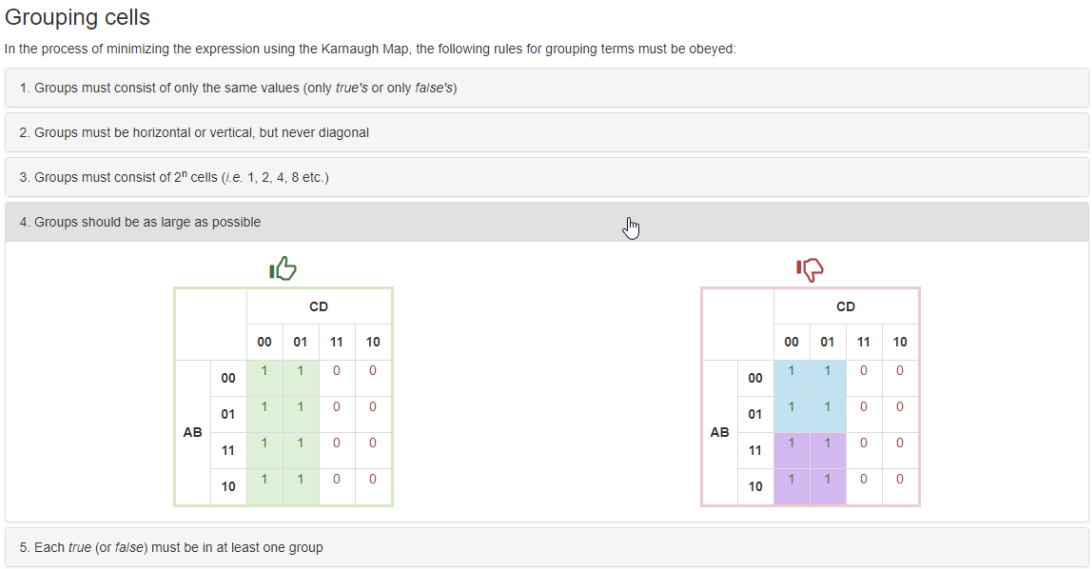

Figure 3.1: Screenshot of the list of grouping rules in the Tutorial page (cropped).

3.1.3 Practice

Practice is the most important section of the application since this is where the user may practise their skills at solving the Karnaugh map related problems and receive instant feedback. The aim of the Practice section is to teach the user to solve the Karnaugh map problems by asking them to complete some examples of them on their own, yet with a slight help. By the principle of observational learning, the user, having seen and attempted exercises themselves, is likely to know how to approach similar problems in future.

The main page of this section informs the user about all available types of exercises with their descriptions. The menu on the side provides a convenient navigation across all exercises. It is displayed the whole time as long as the user is in the Practice section.

All questions are independent on each other and the user does not receive any award for answering them correctly. They may attempt all the questions multiple times, which gives them the freedom of choice and to fail.

Instead of points, the user is presented the feedback which may help them to notice their mistake, which, in turn, implicitly leads them to the solution. Each feedback is accompanied with appropriate colouring of the alerts and question panels. Green indicates that the right answer was provided and red denotes the wrong answer. Yellow colour is also used, specifically when the given answer in only partially correct. It is not always straightforward to point the user's exact error or there may be multiple errors of different types making the screen cluttered with feedback comments. In such cases, colouring provides a satisfactory help. Complete information on the feedback for each exercise type is laid out in Section 3.2 which describes all exercises in detail.

3.1.4 Quiz

Essentially, *Quiz* section consists of exercises of the same types as the Practice section. The difference is that it is designed to test the user's skills rather than to teach them any new methods how to use the Karnaugh map. Gamification of *Quiz* section is supposed to motivate the user to complete all the questions without any help. If they struggle with any task, they may seek guidance by solving similar problems in the Practice section with more informative feedback.

Similarly as in the Practice tab, the main page gives an overview of how the Quiz section works, detailing the rewarding system (described in following paragraphs). Furthermore, there is a side menu with all available exercises which is always visible if in the Quiz section.

Answering questions in this section does not give any feedback other than whether the provided answer is correct or not (green and red colouring is still applied). Instead, every first right answer for a particular question gives the user a number of points that is assigned to that question. To keep the user informed about what questions were already completed, every link to such a question in the side menu turns green. Points are the first step of the gamification of the tool. Not only do they work as positive reinforcement, but they also motivate the user to keep on *playing*. However, just the points are not sufficient.

The progress bar is continuously displayed so that the user can keep track of their progression. The partial progress bar also subconsciously increases the user's need to complete all questions. To intensify that need, badges are introduced. Finishing successfully all questions of the same type results in a badge being rewarded to the user alike completion of the level in a game.

3.1.4.1 Progress Status

The breakdown of the user's results can be found in the *Progress Status* modal window, shown in Figure 3.2. There the user can find the list of all questions in the Quiz grouped by their type. If the user received a badge for completing one section of the quiz, it is displayed next to the exercise type header.

Each row shows how many points the question is worth and how many points the user gained for it. There is also a number of attempts made until the first correct answer was provided as well as total number of how many times the question was attempted. Moreover, the user can reset the question's statistics, which deletes any progress related to the question as if it was never attempted. It can also be done for all questions at once using the *Reset All* button at the top of the window.

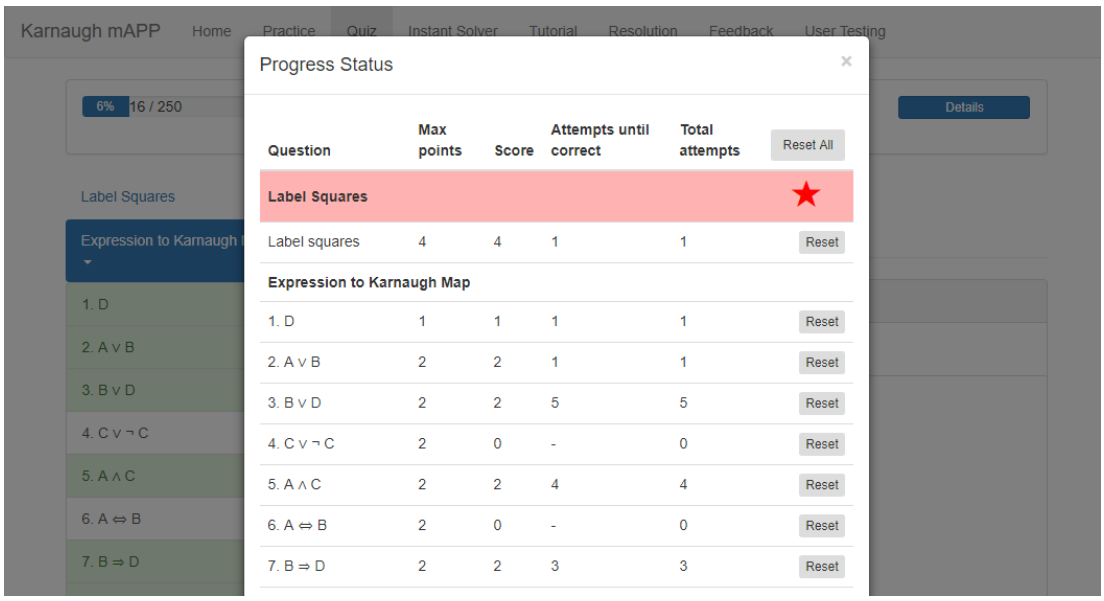

Figure 3.2: Screenshot of the Progress Status modal window with a collected badge for Label Square exercise (cropped).

3.1.5 Instant Solver

Instant Solver is a tool to minimise a Boolean algebra expression into its disjunctive or conjunctive normal form straight away. It lets the user check the solution for any Karnaugh map related problem they might find in textbooks, on the Internet or in past exam papers. The user can either type in an expression they wish to convert or select the states on the Karnaugh map for which the output expression should evaluate to true.

Custom expression can consist of AND, OR, IF-THEN and IFF (if and only if) statements, and may include negations and brackets. Since $\wedge$ and $\vee$ symbols are not available on a typical keyboard, they should be replaced with words ”and” and ”or”, respectively. IF-THEN is represented by ”=>” and IFF by either ”<=>” or ”iff”. The negation symbol $\neg$ is replaced by ”not”. The form validates the input on the fly and disables the *Submit* button if the syntax is invalid. Once the valid expression is submitted, it is shown in the LaTeX-like style to make it more apparent. Also, the states for which the expression evaluates to true are marked on the Karnaugh map.

The instant solver gives an option to choose whether the problem should consider three variables or four. It also lets the user decide if DNF or CNF should be produced. The former finds all the best combinations of groups using all the states for which the initial, hence also the final, evaluates to true. The latter does the same for the states that are excluded.

One expression may have multiple DNFs and CNFs. The *Instant Solver* shows all of them, which is a significant advantage over existing web solvers. Each solution displays all groups as the conjunction of literals satisfying all and only the states in that group. When the group is hovered over, the corresponding states on the Karnaugh

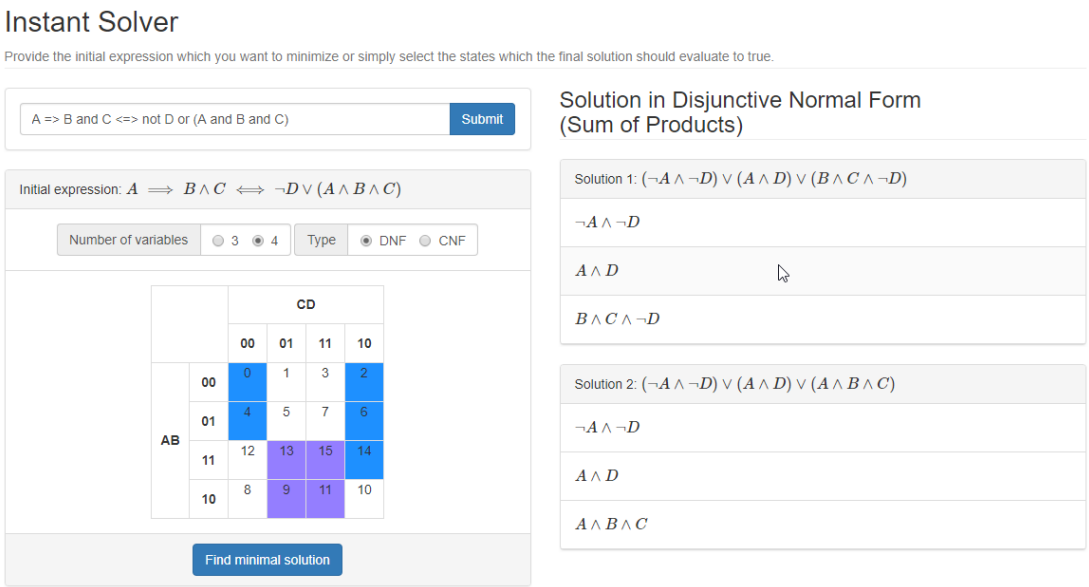

Figure 3.3: Screenshot of the Instant Solver with a custom expression converted to DNF and a highlighted group.

map are highlighted.

Figure 3.3 shows the example of converting the expression ”A => B and C <=> not D or (A and B and C)” into its disjunctive normal form. The expression is displayed as $A \implies B \wedge C \iff \neg D \vee (A \wedge B \wedge C)$. States for which the expression evaluates to true are marked as blue and both minimal DNFs are shown on the side. When the cursor is hovered over the row displaying $A \wedge D$, cells satisfying this group are highlighted.

3.1.6 Resolution

The Karnaugh map provides a nice visualisation of the resolution of expressions with a small number of variables. In fact, it may be used as a smooth introduction to the concept of resolution. *Resolution* tab explains what the resolution and the entailment are, and then describes the correspondence between these concepts and the Karnaugh map in an organised way illustrated with an example. However, this section was not initially planned in this project, thus, its features are limited to presenting a description without any further interaction with the user.

3.1.7 Feedback

Feedback page provides a convenient way for the user to send any feedback about the application. The user can rate it on a scale from 1 to 5 and/or write a comment. The screenshot of this form is presented in Figure 3.4. Upon submission the form input is sent to the server, where it is inserted into a database. This page is the only place on the website where the user communicates with the server excluding the initial fetch

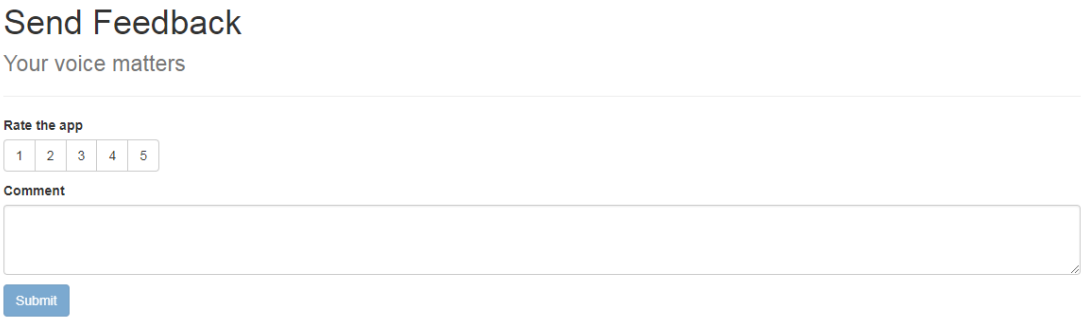

Figure 3.4: Screenshot of the form for the user to send general feedback.

of the application. The detailed process of sending and storing the user's feedback is described in Section 4.7.

3.2 Exercises

The tool is supposed to be a learning aid for a specific course, Informatics 1 - Computation and Logic. Therefore, exercises should be made in alignment with the course requirements.

The tutorial focused on Karnaugh maps in the most recent academic year, namely in October 2017, written by Professor Michael Fourman, involved four kinds of exercises [20]:

1. Label each of the sixteen squares in the Karnaugh map with the corresponding decimal number.

2. Given the expression, mark all the states that are excluded by it. Similarly, the students may be asked to mark all the states, for which the expression evaluates to true.

3. Given the Karnaugh map with marked states on it, find a constraint that eliminates, and an expression that is satisfied by, exactly these states.

4. Given a pair of constraints, use the Karnaugh map to identify a new clause entailed by these constraints. This, in fact, is a combination of two previous tasks.

By combining first second and third tasks, the algorithm for minimising the complex Boolean algebra expression is obtained. However, splitting them into separate exercises makes the usage of the Karnaugh map easier to comprehend. The following section describes all types of exercises included in the tool in more detail.

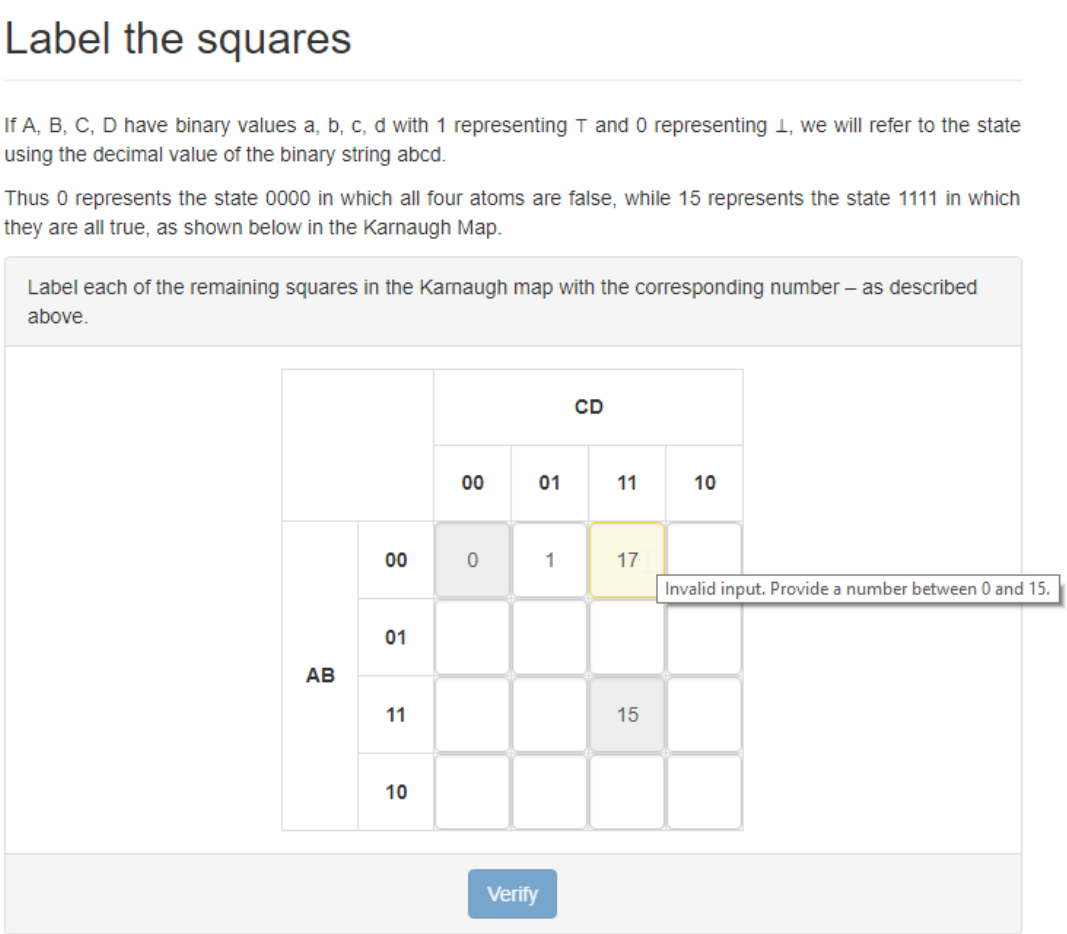

Figure 3.5: Screenshot of Label Squares exercise in Practice section with a warning informing on the range of the possible input.

3.2.1 Label Squares

3.2.1.1 Task

If A, B, C, D have binary values a, b, c, d with 1 representing $\top$ and 0 representing $\bot$, the corresponding state may be referred using the decimal value of the binary string $abcd$. Thus 0 represents the state 0000 in which all four atoms are false, while 15 represents the state 1111 in which they are all true. *Label Squares* exercise consists in labelling squares in the Karnaugh map with a decimal number that corresponds to the binary representation of that state.

3.2.1.2 Objective

This exercise helps to observe that the Karnaugh map holds the same information as the conventional truth table, yet in a different format. Besides, it introduces unique IDs for each cell, which facilitates referring to states in future.

3.2.1.3 Practice Version

In Practice section the introduction to the question includes the right labelling for two extreme states, 0000_2 (0_{10}) and 1111_2 (15_{10}). These states are also pre-labelled in the interactive Karnaugh map. When the user types in the integer outside the range between 0 and 15, the cell changes its colour to yellow and the warning tooltip is displayed, as shown in Figure 3.5. The *Verify* button is disabled until all cells are filled in. Verification in Practice mode gives the feedback on every square separately; the square turns green if the provided number correctly corresponds to the binary representation the state and red otherwise.

3.2.1.4 Quiz Version

The description of the exercise in Quiz section does not contain any examples. The user needs to fill in all the squares themselves. If the entered number is out of range, the warning tooltip in not displayed. Verification in Quiz mode gives the feedback on the answer as a whole, i.e. it assesses the answer as correct provided all the squares are labelled correctly. If at least one of the squares is assigned the wrong value, the entire answer is deemed incorrect and the user is not informed which particular cells contain an error.

3.2.2 Expression to Karnaugh Map

3.2.2.1 Task

The user is shown a Boolean algebra expression and is tasked to mark all the states for which either the expression evaluates to true or that are excluded by the initial expression, depending on the question. If the former is asked, states marked by the user turn green, otherwise they turn red, as shown in Figure 3.6. Such colouring prevents from confusion between two contradictory varieties of the same exercise.

3.2.2.2 Objective

For the Karnaugh map to be useful, the user needs to be able to mark all the relevant states. It is usually the first step to convert a given expression into its DNF or CNF. Besides, it emphasises a tight correspondence between the Karnaugh map and the conventional truth table.

3.2.2.3 Practice Version

The process of marking squares looks the same in both Practice and Quiz sections. The difference is in the feedback given upon the incorrect answer. If the user marks only states which belong to the correct answer, yet does not mark all of them, they are

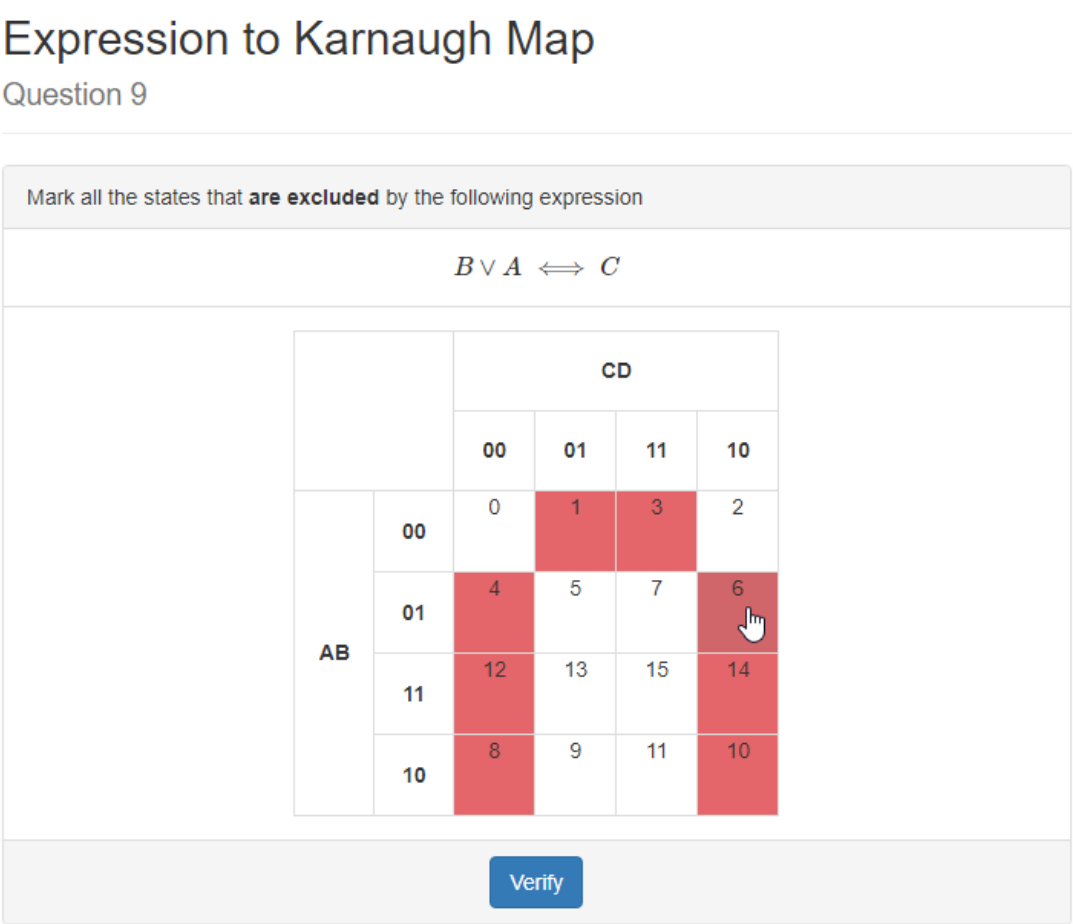

Figure 3.6: Screenshot of Expression to Karnaugh Map exercise in Practice section asking to mark the excluded states.

shown the red alert saying there are more states to be marked. If the user marks all the states that are in the solution, but also marks some irrelevant states, they are informed that too many states were marked. In other case, if the user provides the wrong answer, the detailed feedback is impractical. Then the general feedback prompting to give a question a second thought is displayed. Any time the user is wrong, they can request to see a correct answer.

3.2.2.4 Quiz Version

When the user answers a question in Quiz mode incorrectly, they are asked to attempt the question once again. No detailed feedback is given nor can the user request to see a solution.

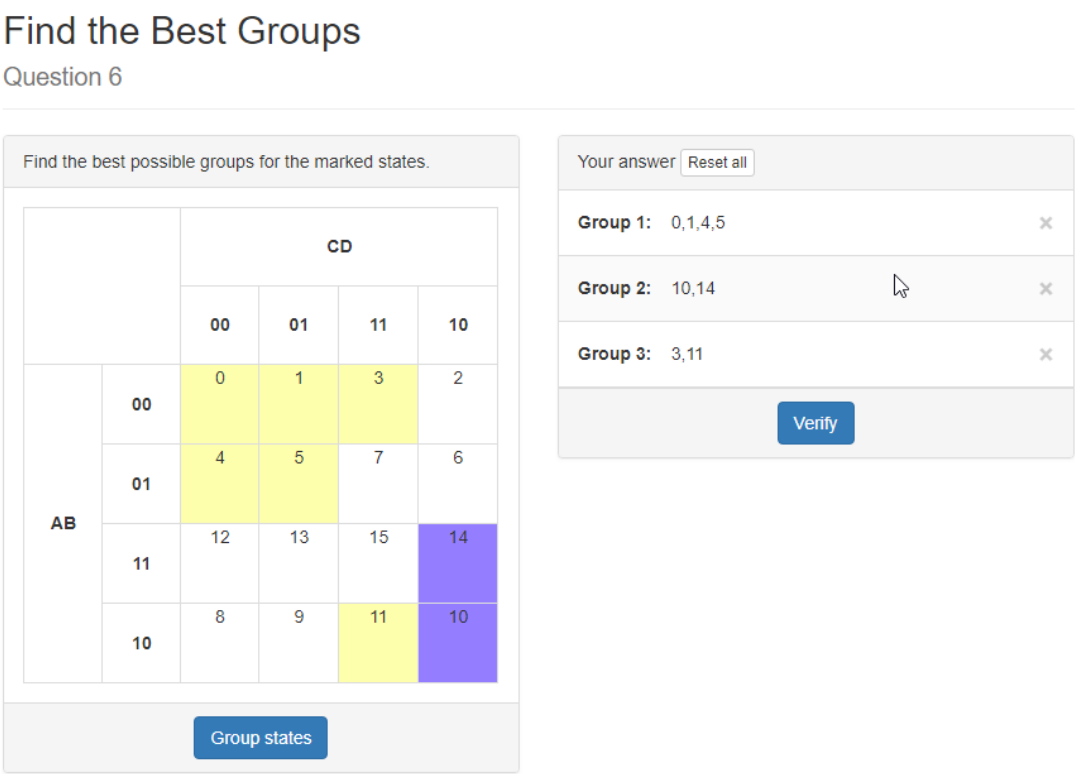

Figure 3.7: Screenshot of Find the Best Groups exercise in Practice section.

3.2.3 Find the Best Groups

3.2.3.1 Task

The user is given an interactive Karnaugh map with pre-marked states which are selectable. They are required to group them according to the rules of grouping states when finding the minimal DNF/CNF, as described in the Section 1.3.1. The user can select states and then group them together by clicking *Group states* button. This adds the group to the user's answer panel, identifying the group by the IDs of the selected cells. The ID is simply the decimal representation of the state symbolised by that cell. When the mouse is hovered over the group row in the user's answer panel, the corresponding group of cells in the Karnaugh map is highlighted. The group can also be removed from the answer by clicking the x icon next to the selected group. Adding an empty group or a group that has already been added results in an error alert. The example of the selected groups by the user is shown in Figure 3.7. If there are more than one solution, any correct answer is satisfying.

3.2.3.2 Objective

The aim of the Karnaugh map is to find the *minimal* form of the initial expression. Grouping cells appropriately is the essential part of this model. Following the rules from Section 1.3.1 ensures the minimal number of groups is found, thus, the final DNF

(or CNF) will consist of as few products (or sums) of literals as possible. This exercise is meant to make the user memorise these rules and know how to apply them. To cover all the rules, some of the questions are more straightforward, whereas others are trickier.

3.2.3.3 Practice Version

Practice mode instantly informs the user if the selected squares do not form a valid group, i.e. they do not make a rectangular block of 1,2,4,8 or 16 cells. Such a group is not added to the user's answer panel. The user is also informed if the added group makes the answer contain a redundant group, i.e. one of the selected groups is the sum of some other already selected groups.

3.2.3.4 Quiz Version

None of the auxiliary feedback from Practice section is displayed in Quiz mode. Consequently, any combination of cells is allowed to be added to the user's answer panel. As always, final verification only shows whether the user found the best groups or not.

3.2.4 Name the Group

3.2.4.1 Task

The user is presented a Karnaugh map with marked states which form a valid group. The task is to pick the correct value for each literal so that their conjunction is satisfied exactly by the marked states. Leaving a literal with no value is equal to selecting the value *Not important*. In such a case the literal does not occur in the expression. When the user pick the value for any literal, the application renders the expression in the LaTeX-like style on the fly.

3.2.4.2 Objective

Grouped states on the Karnaugh map only represent a desired minimal form of an expression graphically. In order to obtain DNF or CNF one must write expressions describing each group. This task involves doing that for one group at a time to make this process clearer.

3.2.4.3 Practice Version

In the Practice mode when the wrong answer is provided, the feedback on each variable is given. If the picked value for a variable is correct, it turns green, otherwise it turns

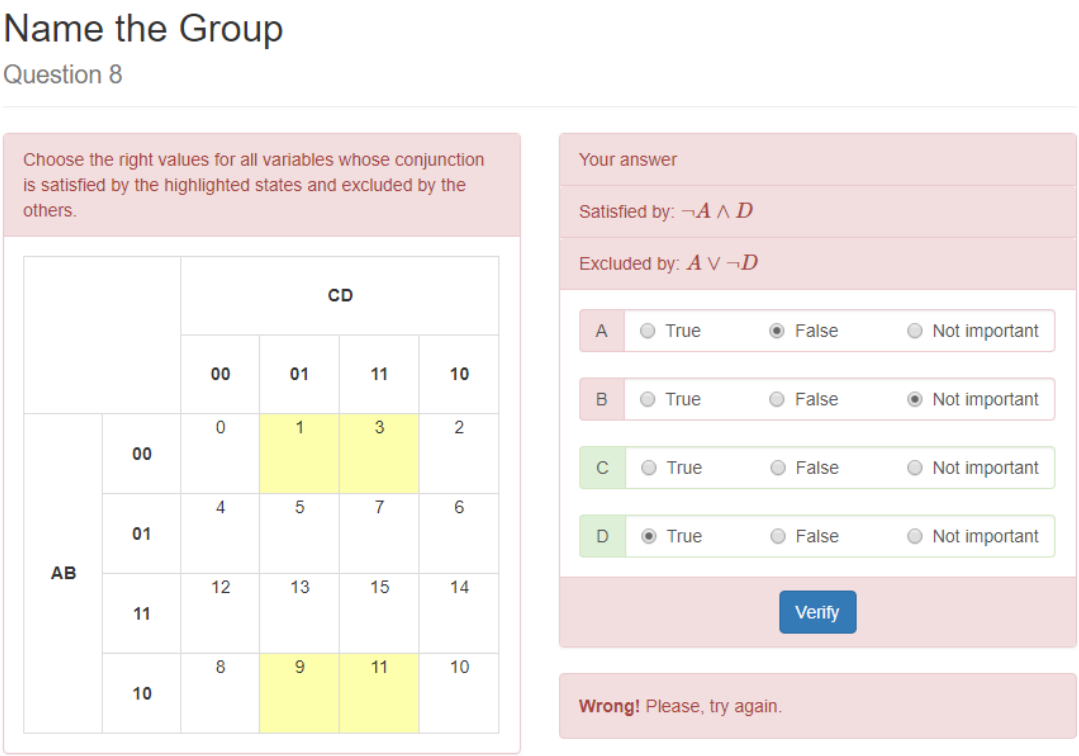

Figure 3.8: Screenshot of Name the Group exercise in Practice section after verifying partially incorrect answer.

red, as shown in Figure 3.8 This gives the user a hint where to look for a mistake to fix without revealing a correct answer.

3.2.4.4 Quiz Version

The Quiz mode does not give any separate feedback on variables. It only indicates the correctness of the answer as a whole. If at least one variable is assigned a wrong value, the answer is deemed incorrect and there is no hint on where the mistake was made.

3.2.5 Karnaugh Map to Expression

3.2.5.1 Task

The user is presented a Karnaugh map with multiple pre-marked cells which they are first required to group as in the *Find the Best Groups* exercise described in Section 3.2.3. For every selected group they are then tasked to assign a value to each variable as it is done in the *Name the Group* exercise discussed in Section 3.2.4. This process produces an expression in disjunctive normal form which is displayed on the fly, as shown in Figure 3.9

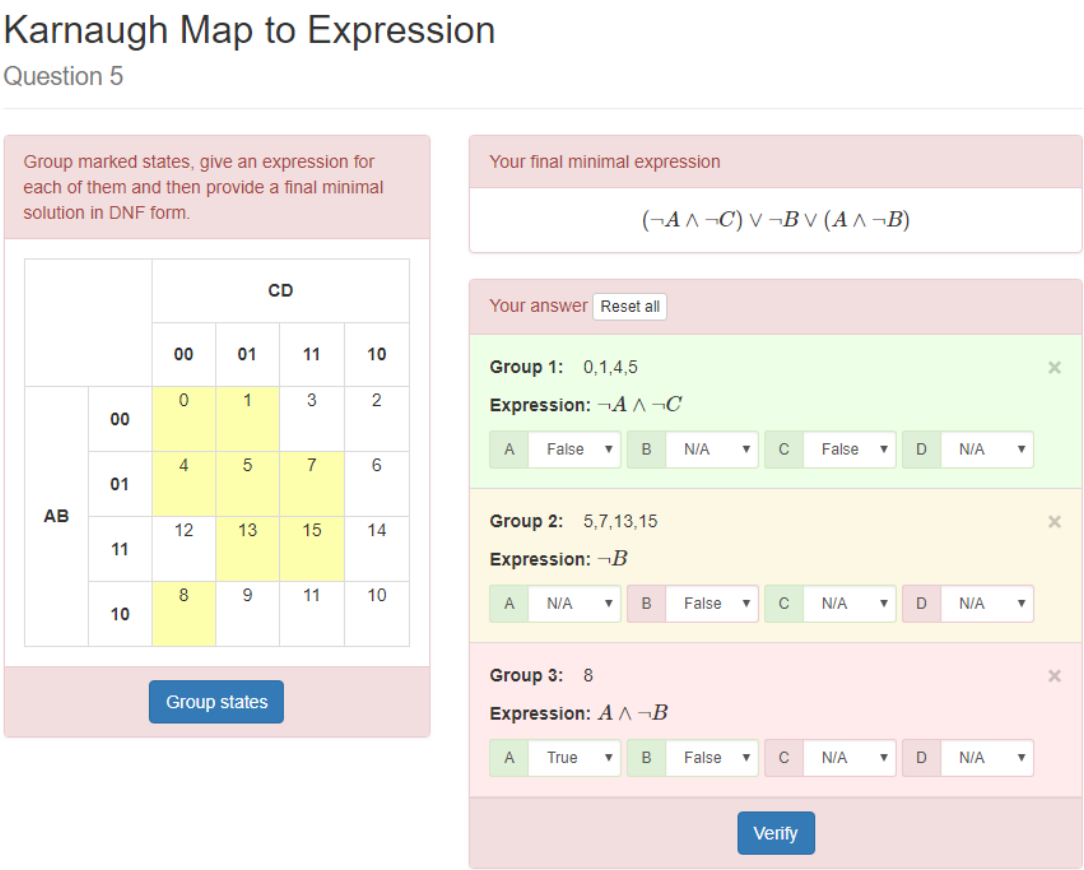

Figure 3.9: Screenshot of *Karnaugh Map to Expression* exercise in Practice section after verifying partially incorrect answer.

3.2.5.2 Objective

Previous exercises focused on unit processes included in the expression minimisation using the Karnaugh map. *Karnaugh Map to Expression* exercise combines two major steps, namely grouping and naming the groups to combine them into a single DNF. This trains the user to incorporate the knowledge from various tasks to solve more complex problems.

3.2.5.3 Practice Version

Some feedback in the *Karnaugh Map to Expression* exercise is taken from two previous exercises which it is based on. Namely, the user is alerted if the selected cells do not form a valid group and is warned if their answer contains a redundant group.

Upon verification every variable in every selected group changes the colour to indicate whether its value is correct for that group, regardless of whether the group itself belongs to the solution. If the selected group is in the right answer and all variables in it are assigned correct values, the group box in the user's answer panel turns green. If the right group is selected, yet not all the variables in it are assigned correct values, the

group box turns yellow. If the group does not belong to the solution and variables were assigned wrong values, the box turns red. Otherwise, the group box remains white. The example of colouring feedback is shown in Figure 3.9. If the user selects all and only the required groups, but not all the variables are assigned correct values, adding and modifying groups is blocked.

One Karnaugh map may be represented by multiple minimal DNFs. In such cases, guessing correctly any of them is sufficient. If the user is wrong, the colouring feedback is given for one of the potential solutions, which the user is informed about. Moreover, they are told how many possible solutions exist.

3.2.5.4 Quiz Version

Feedback in the Quiz mode works as in the previous exercises. Specifically, the user needs to guess correctly *all* the groups in the solution and assign correct values to *all* variables in these groups. Any mistake results in the general negative feedback which does not indicate where the mistake was made.

3.2.6 Minimise the Expression

3.2.6.1 Task

The user is presented a Boolean algebra expression which they must convert into DNF or CNF using the interactive Karnaugh map. First they need to mark all the squares which represent states for which the expression evaluates to true (if DNF is to be produced) or states which are eliminated by the expression (if CNF is to be produced). This process looks the same as in the *Expression to Karnaugh Map* exercise described in Section 3.2.2. Afterwards, the user needs to group the marked states and assign proper values to the variables in each group, as in the *Karnaugh Map to Expression* exercise detailed in the previous section.

3.2.6.2 Objective

This exercise consists of all the steps required in converting the expression into its minimal disjunctive or conjunctive normal form using the Karnaugh map. It comprises virtually all the tasks the user was exposed to in the previous exercises. Therefore, *Minimise the Expression* exercise teaches (or tests) the user to apply all the learned unit procedures to resolve complex problems such as conversion of any expression into its DNF/CNF.

3.2.6.3 Practice Version

In the Practice mode the user can decide themselves whether they want to convert the expression into DNF or CNF. Depending on this setting, either states for which the ex-

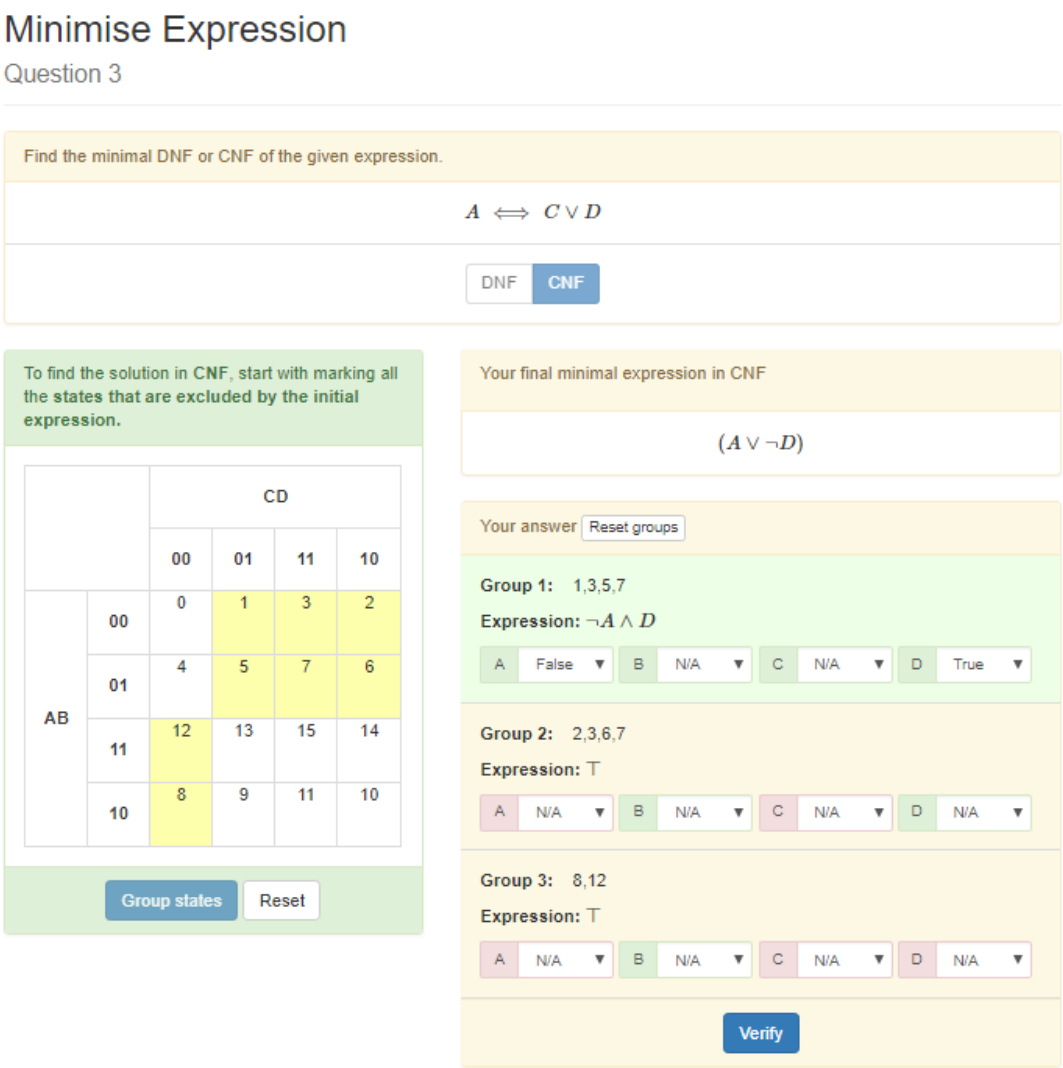

Figure 3.10: Screenshot of *Minimise Expression* exercise in Practice section after verifying partially incorrect answer with the correct grouping but incorrect values of variables.

pression evaluates to true or false should be selected. For this step the same feedback as in the *Expression to Karnaugh Map* exercise (Section 3.2.2.3) is applied. Once the proper states are selected, the Karnaugh map panel turns green and the other states are disabled so that the user can start grouping. From now on, the process and feedback alerts/colouring are the same as in the *Karnaugh Map to Expression* exercise (Section 3.2.5). The final minimal expression formed by the user's answers is displayed on the flye. If CNF mode is selected, the application automatically converts the auxiliary DNF to CNF by applying DeMorgan's laws.

3.2.6.4 Quiz Version

In the Quiz mode it is the question that specifies whether the user is required to find DNF or CNF. The provided feedback is the same as in the previous Quiz mode exercises. Namely, alerts only indicate if the whole answer is correct or not and do not provide details on where the mistake was made. Similarly as in the Practice mode, the user cannot attempt grouping states until all the relevant states are marked.

Chapter 4

Implementation

The source code of the application can be found at:

`https://github.com/Arcadius19/Karnaugh-mAPP`

4.1 Development Tools

4.1.1 Primary Language: TypeScript

The output of this project is an interactive web-based learning tool. This is why the software is mainly developed using TypeScript. It is a typed superset of well-known JavaScript ES6, which is compiled into a plain JavaScript. Consequently, TypeScript includes all the features JavaScript does. Namely, it is a programming language used for programs inserted into the HTML of a Web page, which then are executed by a browser. It enables dynamic interaction with a user by responding to events such as a mouse being clicked or hovered over a particular element of DOM. It is supported by all modern web browser, hence, it does not require a user to install any additional plug-ins or larger software. Moreover, programs written in TypeScript may be easily executed (via browser) on mobile devices.

TypeScript, as an extension of JavaScript, also incorporates elements of object-oriented languages. It introduces type annotations, which facilitates operating on variables of various types and prevents from undesired type coercions. TypeScript's code structure, being based on object-oriented programming languages rather than prototype-based ones, makes it easier to use for those already familiar with Java or C-like languages. It also mitigates the risk of errors caused by the unusual variable scoping[19].

4.1.2 Main Framework: Angular

A framework facilities a process of connecting all components of the application by providing the essential parts of the software upon the project initialisation. The most

popular JavaScript/TypeScript framework is Angular, which is ideal for single-page and client-side applications. Angular 5 is the newest version of the framework at the moment, hence, used in this project. It is written in TypeScript, which was the major reason of selecting this language as the main language of the project.

Angular uses Node.js as the run-time environment, which enables the server to use JavaScript. Furthermore, Angular provides Angular CLI, a utility enabling to manage the project from the command line. It automates tasks like creating projects, adding new components, modules etc. as well as it has a built-in HTTP server that can be used to run the app. This helps to maintain a consistent code across the application [31].

4.1.3 Styling: Bootstrap

The tool is styled using Bootstrap (v3.3.7), which is the most popular front-end web toolkit for designing websites providing a lot of user-friendly feature templates. The main aim of the styling framework is to keep a consistency across the application and improve its maintainability. It reduces the work involved in styling allowing to focus on developing core functionality. Bootstrap's mobile-first approach emphasises responsive web design. Consequently, the app is easy to navigate on screens of any size, from mobile devices through desktops.

4.2 Server: Node.js

To run the Angular app, only a simple static HTML server is required since the entire application is composed on the client-side. Consequently, the server must be configured to return the application's host page (`index.html`) upon a request for a file it does not have [22, deployment]

Node.js is essential for web development with Angular. It is a JavaScript run-time environment allowing the server to execute JavaScript. Since it is already required for Angular and it uses the same language as the entire application, this framework is used for the server-side. It has a built-in module, HTTP, which allows Node.js to transfer data over the Hyper Text Transfer Protocol (HTTP). To facilitate development of the server even further, *Express.js* is used. It is a web application framework for Node.js which provides a set of middleware methods, i.e. functions which execute some code depending on the path the request is sent to.

4.3 Components

One of the main ideas behind Angular is the idea of components. They allow the application to provide the browser with new tags with custom functionality. As the name suggests components are parts that make up the entire application. By splitting it

into smaller units, the code gets more scalable, maintainable and reusable. Examples of components implemented in this project are listed below.

4.3.1 App

The root of the application. It displays the main navigation bar which is visible on every page. The navigation bar keeps track of the active tab based on the current active route.

4.3.2 Interactive Karnaugh Map

It is the primary component in the application as it is the main bridge between the user and the application. Interactive Karnaugh Map component is used in virtually all exercises as well as in the Instant Solver. This component produces an interactive Karnaugh Map-like table. Its cells can be marked and grouped by the user. It also implements methods to (un)highlight some states from an external component, which is used in exercises involving grouping. Since comparing groups selected by the user to the solution groups is a common procedure, a method to do so is also implemented in this component.

4.3.3 Exercises

Every type of the exercises has its super component with a general logic applying to both Practice and Quiz modes. They store necessary variables and implement methods for initialising the component, retrieving question details, populating the interactive Karnaugh map and verifying the user's answer. Every exercise has then separate components for Practice and Quiz sections which extend the corresponding super component. The subcomponents may add methods and variables or modify existing ones so that they support required functionality for a specific mode, such as relevant feedback or rewarding with points. Components can only extend the behind-the-scenes logic, therefore, each child component has its own HTML template. Templates for two different modes of the same exercise differ mainly by the extent of feedback messages.

Component inheritance supports the DRY principle. DRY stands for *don't repeat yourself* and, as the name suggests, is the software engineering principle which aims to reduce repetitions in the software by replacing repetitive parts with an abstraction. This decreases a number of lines of the code and improves maintainability by reducing a number of places in the code that need to be modified if some logic is changed.

4.3.4 Practice

This component is the root component for the Practice section. It displays the side menu with all the practice questions. Next to the menu a child component is inserted.

The child component is either introduction component which displays general information about the Practice section or it may be a relevant question component, as described in Section 4.3.3.

4.3.5 Quiz

The Quiz component works similarly as the Practice component, yet it involves a different set of questions. Another difference is that the Quiz component displays the user's progress bar throughout the Quiz section. The progress bar also contains a button opening the Progress Status modal window which is a separate component described below.

4.3.6 Progress Status

The component represents the modal window showing the user's progress statistics in the Quiz section. It is fed with the information about the exercises by the parent component, Quiz Component (discussed above). It also enables resetting statistics for one or all questions. To do so, it is allowed to read and update the local storage where the statistics are stored. The storage of the user's statistics is detailed in Section 4.6.

4.3.7 Home

It simply displays the information on the Home page and does not contain any logic behind.

4.3.8 Tutorial and Resolution

Both components present static pages describing either Karnaugh map or the resolution procedure. Since they do not involve any interactive tasks, there is no logic in them other than to simplify producing the Karnaugh map-like HTML table.

4.4 Exercise Distributors: Services

Service is a singleton object with a narrow, well-defined purpose to separate some logic from a component's view-related functionality. It is commonly used to pass data across the application, including the server. Therefore, a service class might provide streams of data that the application can subscribe to or provide methods to add or modify data [22, Introduction to services and dependency injection].

In this application every type of the exercises has its own service which distributes questions of that type for both Practice and Quiz modes separately. Each of these

services extends a super class, `ExerciseService`, which outlines a general structure of the service and implements methods used by all sub-services.

For every type of questions there is also one exercise class which extends the super class, `Exercise`. All questions of one type are instances of that exercise class. In turn, every exercise service stores two arrays of practice and quiz questions of the type corresponding to that service. Moreover, they all have 8 main methods to retrieve question(s) in various configurations. Every method implements one setting from each of the following pairs:

- Retrieve question(s) from either Practice or Quiz question sets
- Retrieve either all questions or one question with a specific ID
- Retrieve questions(s) synchronously or asynchronously

The current implementation does not require asynchronous retrieval since the questions are stored in static arrays. However, in time one might want to fetch the questions dynamically from the server, for which asynchronous requests would be desired. Asynchronous retrieval of the question also enables reusability of the exercise component when a new question of the same type is loaded.

Services are also responsible for updating the user's statistics when attempting the exercise in the Quiz section. Hence, they directly access the local storage. Details on the storage of the user's progress statistics are described in Section 4.6.

4.4.1 Exercise Classes

As mentioned above, all questions are instances of some sub-exercise class which extends super `Exercise` class. Regardless of the type, each question has following properties:

- **id:** unique integer in a set of questions of the same type and the same mode
- **name:** string representing the question which is displayed in the browser (e.g. "$A \wedge B$" or "Set 1")
- **completed:** Boolean value indicating whether the question has already been successfully completed by the user
- **points:** number of points the user gains for answering the question correctly, if the question is in Practice mode this property remains `null`

Naturally, every sub-exercise adds some properties relevant to its type. These are as follows:

Expression to Karnaugh Map

- **expression:** string which is displayed to the user and describes the states that should be marked or left unmarked (syntax of the string is the same as the input in the Instant Solver described in Section 3.1.5)

Find the Best Groups

- **nVars (3 or 4):** number of variables in the Karnaugh map; 3 will produce the Karnaugh map of dimensions 2x4 whereas 4 will display 4x4
- **cells:** array of cell IDs which are premarked, thus, to be grouped by the user (cell ID is the decimal representation of the state)

Name the Group

- **expressionGroup:** an instance of `ExpressionGroup` class which has four properties, each representing one variable (see Section 4.5.1.1). Every property takes a Boolean value which is a correct answer. If the variable is assigned `null`, its value changes across the group, thus, is not important. The component for this type of the questions uses this property to properly pre-mark the Karnaugh map shown to the user

Karnaugh Map to Expression

- **cells:** same as in *Find the Best Groups* exercise

Minimise the Expression

- **expression:** same as in *Expression to Karnaugh Map* exercise

4.5 Auxiliary Classes

Main components and services described so far alone would not be sufficient to provide full functionality or high maintainability. This section discusses miscellaneous classes used in this project that are important for the tool to work properly.

4.5.1 Group Representation

The Karnaugh map model involves grouping and then naming the groups. In this context a group is a set of literals whose conjunction or disjunction represents a set of states in the Karnaugh map. In the project there is a lot of operations on them. Implementation of some of these operations are easier using different representations of the group than the others, therefore, there are various ways to represent one group.

4.5.1.1 ExpressionGroup class

Class having four properties (`aVar, bVar, cVar, dVar`), each of a Boolean type. If a variable takes `null` value, its value changes across the group, so it does not identify it. The object is usually treated as a conjunction (AND) of these literals, yet it may not always be this case when converting the object into the LaTeX-form expression.

As an example:

```
ExpressionGroup{aVar: true, bVar: null,
                cVar: false, dVar: null}
```

represents $A \wedge \neg C$, but could be also used to display $A \vee \neg C$

4.5.1.2 GridGroup class

Class representing the block in the Karnaugh Map with coordinates of the relevant states. The coordinates indicate the top right corner of the block and the number of rows and columns spanning from that corner. It needs to be noted that the origin corner could have larger indices than the corners on the bottom or the left side since the opposite edges are actually adjacent and the group could, for example, have its right edge on the left side of the map and its left edge on the right side of the map. `GridGroup` class has four properties:

offRow: Index of the the upper row of the block (zero-based numbering)

offCol: Index of the right column of the block (zero-based numbering)

rangeRow: Number of rows the group spans across

rangeCol: Number of rows the group spans across

As an example:

```
GridGroup{
    offRow: 1,
    offCol: 3,
    rangeRow: 2,
    rangeCol: 2
}
```

would describe the following group in the Karnaugh map:

		CD			
		00	01	11	10
AB	00				
	01				
	11				
	10				

4.5.1.3 Array of Cells

Array of integers, each of which is the ID of the cell belonging to the group. The cell ID is a decimal representation of the state corresponding to that cell.

As an example, an array `[3, 7, 11, 15]` represents the following group:

		CD			
		00	01	11	10
AB	00	0	1	3	2
	01	4	5	7	8
	11	12	13	15	14
	10	8	9	11	10

4.5.1.4 Karnaugh Map-like Array of Arrays

Array of binary arrays where each unit element represents one cell. Each element takes a value of either 0 or 1 depending on whether it is unmarked or marked, respectively. This representation is often called `map` in descriptions of some methods.

As an example,

```
[[0, 0, 1, 1],
 [0, 0, 1, 1],
 [0, 0, 1, 1],
 [0, 0, 1, 1]]
```

corresponds to the Karnaugh map shown below:

		CD			
		00	01	11	10
AB	00				
	01				
	11				
	10				

4.5.2 Best Group Solver

`BestGroupSolver` is a core class in the project. It is used to find the best groups for a given Karnaugh map following the procedure described in Section 1.3.1. Its main method is *`findBestGroups`*`(markedArray: `**`number[][]`**`, dnfType: `**`boolean`**`)`. The argument `markedArray` is a Karnaugh map represented as an array of arrays as described in Section 4.5.1.4, where every element equal to 1 means that the initial (and final) expression is true for the corresponding state and element

equal to 0 means the corresponding state is excluded by that expression. The argument `dnfType` indicates in which form the final minimal expression is supposed to be. If `dnfType` is `true`, which is a default value, the sought expression is in disjunctive normal form, so the algorithm groups states marked as 1. If `dnfType` is `false`, the sought expression is in conjunctive normal form, so the algorithm groups states marked as 0. The method returns an array of arrays of `ExpressionGroup` objects, where each array is one possible solution. Every solution should be treated as in DNF, i.e. a disjunction of groups it consists of and each of these groups is a conjunction of the literals. It should be noted that this format of the solution also applies when final desired form is CNF. This is because `BestGroupSolver` is responsible for grouping not computing the final expression. If one wants to obtain CNF, the solution array (treated as in DNF) would have to be negated.

The algorithm goes over all possible groups in the Karnaugh map (81 groups for 4 variables) starting from to the largest. It checks whether all states in the group belong to the set of relevant states in `markedArray`. If so, the group is added to `bestGroups` array. After checking all the groups of one size, all states included in any of the found groups are marked as already checked so that the smaller group is not added if it contains all states that are already in a larger suitable group. Having gone through all the potential groups, the algorithm finds all minimal combinations of groups in `bestGroups` array that cover all relevant states from `markedArray`. Final groups are transformed into `ExpressionGroup` form.

4.5.3 Karnaugh Map

`KarnaughMap` class is an abstract representation of the Karnaugh map. Note the class itself is not abstract. It can represent the Karnaugh map either with 3 or 4 (default) variables. It stores an array of cell IDs and flags used to get a value of some variable for a given state. As an example, for 4 variables, a state can be presented as *abcd* which is some binary number. The ID of the corresponding cell is a decimal representation of that number. There are four flags, for each variable. For a variable A, the flag is `1000` (8_{10}) since a bitwise AND operation with the cell ID (binary representation makes it easier to visualise) yields the value of A. Table 4.1 shows the complete difference in properties of `KarnaughMap` objects with 3 and 4 variables.

`KarnaughMap` class implements various conversion methods, such as:

- `cellToMap()`: array of cell IDs to array of arrays (map representation)
- `mapToCells()`: array of arrays (map) to array of cell IDs
- `mapToExpression()`: array of arrays (map) to `ExpressionGroup` instance
- `expressionGroupToCells()`: `ExpressionGroup` object to array of cell IDs

Nevertheless, the most important method of the class is *`evaluate`*`(query: `**`string`**`)` which takes an expression in a special format, as described in Section 3.1.5 and returns an array of arrays of integers (map) where each unit element (cell) takes value of 1 or 0 depending whether the corresponding state satisfies the expression or

number of variables	3	4

`cellIDs` array	`[[0, 1, 3, 2],`
	`[4, 5, 7, 6]]`
		`[[ 0, 1, 3, 2],`
		`[ 4, 5, 7, 6],`
		`[12, 13, 15, 14],`
		`[ 8, 9, 11, 10]]`

flags	flagA = 4 (100_2) flagB = 2 (010_2) flagC = 1 (001_2) flagD = undefined	flagA = 8 (1000_2) flagB = 4 (0100_2) flagC = 2 (0010_2) flagD = 1 (0001_2)

Table 4.1: Comparison of properties of two KarnaughMap objects with different numbers of variables.

not, respectively. It first parses the expression converting it into the `Expression` object. This process is detailed in Section 4.5.4. The instance of `Expression` class may evaluate itself given an assignment of variables included in that expression. Therefore, to evaluate the initial expression, the algorithm traverses through all the cells in the Karnaugh map and evaluate the expression for an assignment of variables in the corresponding state. The values of variables are computed using the cell ID and variable flags, shown in Table 4.1. If the expression evaluates to true the corresponding cell in the result array is set to 1, otherwise to 0.

4.5.4 Expression Parser

The important aspect of the application is the evaluation of an expression for various states, i.e. various combinations of literals. The program needs to be able to take a custom expression and substitute variables in it (A,B,C,D) with different values and evaluate it with these values. The simplest way to do that would be using the built-in `eval()` function. One could write an expression as a string using JavaScript syntax, i.e. `!!` as $\wedge$, `||` as $\vee$ and `!` as $\neg$, then replace all variables with `true/1` or `false/0` and `eval()` the new string. There are two significant drawbacks of this approach:

- When providing a custom expression in the instant solver, the user could inject malicious code which could have harmful consequences when executed automatically
- There are no basic operators in JavaScript for IF-THEN and IFF statements

Therefore, a *parser*, which would first sanitise a provided expression, then parse it and finally evaluate it for all the states, is desired. For this purpose, the external library, *expr-eval*, is used [3]. It has built-in support for a lot of mathematical operations, however, these are not relevant to evaluating logical expressions, hence, options supporting them are disabled. The options enabled in the constructor of a global `Parser` object are `logical` and `conditional` which, combined, support AND, OR, NOT and ?: (conditional operator).

The parser in the library does not support IF-THEN and IFF operations which are widely used in logical expressions. Therefore, two custom methods are added, namely `iffParse(query: string)` and `ifThenParse(query: string)`. They both work in a similar manner. They scan an argument string and when `<=>/iff` or `=>` is found, they extract four parts of the expression:

- α: part of the expression which precedes the IFF / IF-THEN statement
- β: part of the IFF / IF-THEN statement on the left hand side of the operator
- γ: part of the IFF / IF-THEN statement on the right hand side of the operator
- δ: part of the expression which follows the IFF / IF-THEN statement

Subsequently, `iffParse(query: string)` converts `(β <=> γ)` to `(((β) and (γ)) or (not (β) and not (γ)))` and `ifThenParse(query: string)` converts `(β => γ)` to `(not (β) or (γ))`. The converted string is then inserted in place of the initial statement.

As an example, the expression

```
A or (B or D <=> C) and D
```

would be converted into

```
A or
(((B or D) and (C))
or
(not (B or D) and not (C)))
and D
```

The above methods are executed in `preParse()` method which is run before the *expr-eval* library function `parse()`. This method can handle AND, OR and NOT so no further conversion is required. In `preParse()`, `iffParse()` is run before `ifThenParse()` to preserve a correct logical operator precedence.

4.5.5 LaTeX-style Text

The user inputs an expression using a special syntax (described in Section 3.1.5) which makes it feasible to type in with a typical keyboard. However, this makes it cluttered, thus, harder to read. Displaying all the expressions (user inputs, questions and answers) in the LaTeX format enhances the user interface, which increases general user experience.

MathJax is a JavaScript library which provides a display engine for LaTeX notation in all modern browsers, i.e. it can translate LaTeX commands into high-quality and professional-looking text [4]. Normally, LaTeX formatted text is produced by writing an appropriate command inside special delimiters. Namely, $$...$$ and \[...\] for displayed mathematics, and \(...\) for in-line mathematics. Nevertheless, all logical

expressions that are converted into LaTeX form are produced dynamically and Angular displays them straight away not having them first processed by MathJax. Therefore, one needs to call MathJax to process a text when it is provided. This operation is performed in multiple places across the application, so a custom *directive*, called `MathJaxDirective`, was created.

Directive is a custom HTML attribute which can change the appearance or behaviour of a DOM element [22, Attribute Directives]. In fact, the already seen component is one, and the most common, kind of directives. Creating a new component for handling MathJax text would be overkill, so an *attribute directive* is used. Attribute directives are employed as attributes of elements.

`MathJaxDirective` needs to store a text to be displayed and the text needs to be available for binding. Hence, the directive has an input property `mathText`. When its value changes, it is inserted into the DOM element to which the directive was applied. Then, MathJax is called to process a content of this element.

To facilitate translating a custom syntax expression into LaTeX code, `MathJaxConverter` class was created. It implements *`toMathJax`*`(expression: `**`string`**`)` method which substitutes all operators represented as normal text, e.g. `and`, `not`, with corresponding LaTeX commands, e.g. `\land`, `\neg`. Lastly, the converted string is enclosed with \(...\) to notify MathJax that the returned string should be displayed as an in-line LaTeX text.

`MathJaxConverter` also provides a method which converts a string to a form which the browser could decode, named *`toBrowserText`*`(expression: `**`string`**`)`. It works in the same manner as `toMathJax()` method, yet it substitutes operators with equivalent character entities. As an example, `and` is changed to `∧` and `<=>` is changed to `⇔`. This method is used to display expressions in question menus in Practice and Quiz sections.

4.6 Client-Side Data Storage

The application needs to have the ability to store data regarding the user's progress in Quiz. In general, information on the user could be stored either on the server or the user's browser.

An HTTP cookie (web cookie, browser cookie) is the most common tool to store data on the server. The cookie is a small piece of non-executable data which the server sends to the user's web browser where it is then stored. The cookie is sent back to the server upon every request so that the server can identify from whom the request was sent and retrieve information about this sender. This way, stateful sessions using stateless HTTP protocol are possible. Nevertheless, in this application the following disadvantages of cookies may be identified [30, HTTP cookies]:

- The virtually entire website is client-side, so it does not need to communicate with the server after the initial retrieval. Implementing cookies would require redundant communication with the server.

- The user can often clear cookies, yet the data on the server that is associated with them persists, which would unnecessarily clutter the server.
- People may be concerned with cookies as they are often deemed to violate the user's privacy. Although no sensitive data is stored in this application, one may wonder whether the server tracks the user's activity and shares it with other websites through third-party cookies.
- According to the EU directive from 2011, a website willing to store or retrieve any information from the user via cookies needs to first receive the user's explicit consent to do so [18]. A request for such a permission may worry the user and even discourage them from using the application. The information stored as part of the Quiz is not so important to trouble the user.

For the purpose of this application, web storage is more practical. It allows to store key-value pairs locally within the user's browser. There are two mechanisms within web storage [30, Web Storage API]:

sessionStorage
: stores data for the duration of the session, i.e. as long as the browser is open

localStorage
: stores data permanently, i.e. with no expiration date

To let the user complete all Quiz questions at any time they want, with breaks in the meantime, `localStorage` is more sensible. Therefore, this kind of storage was implemented in this project.

When the user attempts a question in the Quiz section for the first time, new entry in `localStorage` is created. The entry is of the following form:

```
qKey: {
    points: number,
    attemptsUntilCorrect: number,
    attempt: number
}
```

`qKey` is the entry key. It is a global unique id of a question of a form: `"ex" +` **`exID`** `+ "q" +` **`qID`** where `exID` is the id of the exercise type and `qID` is the id of the question within that type of exercises. If there is only one question of a particular type of exercises, like in *Label Squares* exercise, its `qID` is 1.

`points` property specifies how many points the question is worth. Self-descriptive `attemptsUntilCorrect` indicates how many attempts the user made until they answered the question correctly for the first time and `attempts` states the total number of attempts (including those after successful completion of the question). The question entry is updated on every attempt irrespective of the result.

It must be noted that the web storage can only store strings, thus, the above JSON object needs to be stringified before writing it into the storage or parsed after reading.

`localStorage` also stores a total number of points (`totalPoints`) the user has gained. This field is used in the progress bar in the Quiz section. It is updated whenever the user answers a question correctly for the first time.

Data in `localStorage` may be deleted by the user directly in the browser's options, but they can also do that via Progress Status window. For every question there is a button which, when pressed, deletes a corresponding entry from `localStorage`. At the top of the window there is also an option to delete all entries.

4.7 Sending Feedback

Everything described so far is fully executed on the client-side. This is how the Angular application works. When the user requests anything from the server corresponding to the application's domain, the server returns the application's host page. The application is then executed by the browser handling the actual request. It works well provided the user does not need to operate on (CRUD: Create, Read, Update, Delete) any particular data from the server (not included in the application's code or dynamically modified on the server-side). When the user sends feedback, it needs to be sent to the server so it can be read by the actual addressee. Therefore, Feedback page (introduced in Section 3.1.7) is the only place in the application where the user communicates with the server.

4.7.1 Communication with Server: ContactService

For this purpose, a special service, `ContactService`, was created. It is responsible for a direct communication with the server so that it is separated from the component's logic. The service implements a method *`sendFeedback`*`(feedback:` **`Feedback`**`)` which returns an Observable. Firstly, `Feedback` is a class representing feedback the user may send, which consists of two properties, `rating` and `comment`. Secondly, Observable is a type implemented by RxJS (Reactive Extensions for JavaScript) library used by Angular. So-called observer can subscribe to an Observable so that it reacts to whatever the Observable emits. This supports concurrent operations since the observer does not block while waiting for the Observer, but instead it stands ready and reacts appropriately once the Observable emits an object [6][22, Observables]. In general, calls to the server are asynchronous so that the application does not freeze while waiting for the response across the Internet. Therefore, the Observable type substantially facilitates handling communication with the back-end service.

Specifically, the function `sendFeedback()` returns the result of other function provided by Angular's `HttpClient` module. It offers a simplified client HTTP API based on the `XMLHttpRequest` interface supported by browsers [22, HttpClient]. The returned function is `post()` which sends a POST request to the path `'/api/feedback'` on the server with a stringified `feedback` object in its body. The method returns an Observable which the service subscribes to upon the submission of the Feedback form. The Observable will inform the service whether the POST request was successful or

not, and the Feedback component will display an appropriate alert depending on the result.

4.7.2 Handle the HTTP Request: Routing

The server uses `Express` module to implement a route for the POST request to the specified path `'/api/feedback'`. Generally, routing methods have access to the request object and response object [34]. Therefore, this routing function reads the request body and extracts `rating` and `comment`. If they are both empty, the server sends an error message using `response` object. Otherwise, it inserts the feedback into the database.

4.7.3 Storing Feedback: PostgreSQL Database

This application uses PostgreSQL, which is an object-relational database management system [9]. It was chosen mainly because of the prior familiarity with the system. It provides an organised way to store data as entries in Tables (relations). For the purpose of storing the user's feedback, Table *Feedback* was created. It has four columns:

- **id:** automatically incremented integer
- **submissionTime:** timestamp of the submission
- **rating:** integer indicating the user's rating of the application (if any)
- **comment:** the comment the user left in the form (if any)

As mentioned in the previous section, the properties `rating` and `comment` are extracted from the HTTP request body. Using `pg` module [5], the server connects to the database, sanitises the user's input (to prevent from the SQL injection) and inserts a new entry into the `Feedback` table. If successful, the server returns the `feeback` object it received from the client (user). Otherwise, it sends a response with an error status and message.

Chapter 5

Evaluation

5.1 Information Systems Success Model

Information system is a computer-based system which provides information to users in one or more organizations. However, the term could be even broadened to include all computer-based systems [10]. William H. DeLone and Ephraim R. McLean created a framework, called information systems success model (alternatively IS success model), which breaks the IT-system down into six factors to effectively assess its success [13, 39]. The factors should be treated interrelated rather than independent as they may considerably affect each other. These success dimensions are:

Information quality
: Examines the accuracy of the information the system presents and if it conveys the intended meaning.

System quality
: Considers a purely technical aspect of the system, e.g. whether it is functional, reliable, efficient and easy to use.

Service quality
: Assesses the service offered by the service, but also by the people behind the system, e.g. the customer support.

Intention to use (attitude) / Use (behaviour)
: Inspects the user's willingness to use the system, e.g. by frequency of use or time of use, considering whether the use was voluntary or imposed.

User satisfaction
: Measures the user's general opinion on the system, analysing the user experience.

Net benefits
: Explores the impact the system has in general, e.g. on the user, society, industry or organisation (investigated fields should be chosen depending on the system).

`Karnaugh mAPP` is an example of the information system and the above framework can be used to evaluate the project result to some extent. Since all factors depend on each other, they will not be discussed one by one, but instead all described methods of testing will refer to some dimensions.

5.2 Ensuring High Information and Service Quality

The project's aim was to develop a learning tool for a specific course, namely Informatics 1 - Computation and Logic at the University of Edinburgh. Therefore, all the information included in the application should be aligned with the course material and potentially extend it.

All the exercises from the recent tutorial sheet were implemented in the tool [20]. This applies to *Label Squares, Expression To Karnaugh Map, Name the Group, Karnaugh Map to Expression*. The latter was even extended to questions involving more than one group. The application also provides an exercise for the full minimisation of the expression from any expression displayed as a text to its minimal disjunctive or conjunctive normal form using the Karnaugh map. Basing the application's content on the course's syllabus ensures it meets the user's expectation about the information they can find here. This improves its **information** and **service quality**.

The service quality also depends on whether the questions are constructed in a way that the user actually learns anything by completing them. Therefore, some features following the principles of learning, detailed in Section 2.1, were implemented. The user has a chance to see a grouping process step by step supported with visual examples. This works as a model in the **observational learning**, which teaches the user how to approach similar problems they may encounter. Besides, the instant solver and numerous practice questions provide a solid reference for similar tasks in future.

Correct answers in Quiz are rewarded with points, which then are converted into badges. This gamification technique acts as the **positive reinforcement**, which is one of the strategies in **learning by association**, particularly **operant conditioning**. It encourages the user to learn the concept of the Karnaugh to be finally able to answer the question.

The information and system qualities are also enhanced by giving the user a chance to send any **feedback** directly from the website. If they find any inaccuracy or flaw in the application, they may send an informative comment so that the error can be investigated and fixed. This facilitates continuous maintenance of the system.

5.3 Unit Testing

Functional testing checks whether the software meets all the requirements included in the specification, thus, it is important measurement of **system quality**. In general,

not much emphasis was put on functional testing as developing tests covering the system's entire functionality would consume a substantial part of the whole project, which would result in a very limited number of features. However, some functions are critical for the application and should be ensured to work properly.

Unit testing is one of the functional testing methods, which allows to test the correctness of individual classes and methods separately from the rest of the system. Therefore, unit tests were created to guarantee the essential methods were implemented correctly. The default (installed automatically with Angular CLI) testing framework for Angular is Jasmine, which is a development framework for testing JavaScript code independently on any other JavaScript framework. Tests are run with Karma test runner. Karma executes source code against test code for each of the connected browsers and the results are examined and displayed via the command line. Chrome browser also displays the test output in the user-friendly form using *Jasmine HTML Reporter* [22, Testing][35, 42].

Examples of methods that were decided to be tested:

- `findBestGroups()` in `BestGroupSolver` class - for a given Karnaugh map (represented as an arrays of arrays of binary values), return all possible minimal expressions in DNF as an array of arrays of `ExpressionGroup` objects (see Section 4.5.2)
- `findMinimal()` in `ExpressionGroup` class - for an array of `ExpressionGroup` objects (treated as one expression in DNF), find all possible arrays representing the same expression with a minimal number of elements; in other words, minimise the expression represented in DNF
- `mapToExpressionGroup()` in `KarnaughMap` class - convert a map (array of arrays of binary values) into `ExpressionGroup` object; first validate if the cells marked in the map can form a valid group

First of them is the most important and implicitly involves the others, as it was already detailed in Section 4.5.2. One of the tests for this method is shown in Appendix A.1. It detected a major fault in the method's code. Not only did the previous version return only one solution (even if more were possible), but it also sometimes returned more groups than necessary, which violates the Karnaugh map model's principle. The same fault can be found in some of the existing tools discussed in Section 1.4. The problem lay in marking states which were already checked and included in some group. Namely, once a proper group was found, all states belonging to that group were labelled as already-checked instantly. It sometimes led to the situation that the function completely discarded a group which consisted of states that were included in other already found groups. However, this group might be used in a different, also appropriate, solution. The fault was fixed by changing the place where states included in some solution group are marked. Now it is done only after checking all potential groups of the same size. Then, `ExpressionGroup.findMinimal()` is called to find all possible minimal groups of `ExpressionGroups` objects to cover the same space in the Karnaugh map.

The application was also repeatedly tested manually by completing various scenarios

directly in the browser. The behaviour of functions, values of passed arguments and how they are modified were analysed with `console.log()` method which prints its argument in the browser console.

Beyond the system quality, any kind of testing also improves the **information** and **service quality** since it ensures the application delivers the appropriate and accurate content to the user, aligned with their expectations.

5.4 Analysis of ”Intention to Use/Use” Factor

In order to motivate students to use the application, some **gamification** characteristics (see Section 2.2) were introduced. Questions are split into two section, Practice and Quiz. In the former the user can attempt them in an uncontrolled manner having a freedom to fail. Moreover, if they fail, they are presented feedback to guide them to the correct answer. The latter section is reminiscent of a game, where the user gains points for every right answer. These are later exchanged into badges and the user's goal is to collect all of them. The user's progress is continuously displayed in the progress bar, which stimulates their need to complete.

The software does not implement any tool to measure its usage frequency. Nevertheless, even if such a tool was implemented, it would be difficult to measure actual usage of the website now since the course is delivered only in the first semester of the academic year, i.e. between September and December.

The application already provides a service, `ContactService`, which enables a direct communication with the server. It could be extended so that it passes information on every attempt of any question, e.g. to keep track of the general usage or questions with which the users struggle the most.

The usage of the application could be encouraged or requested ahead of the tutorial concerning the Karnaugh map. However, one must bear in mind that enforced usage may result in lower user satisfaction. Moreover, as mentioned in Section 2.2.1, if the user deems the application controlled or enforced, it may not be effective in teaching.

5.5 Usability Testing

User satisfaction may only be measured with actual users. Therefore, usability testing was employed. It consists in creating various scenarios and analysing the user's behaviour as they perform task(s) included in a scenario. Such testing may be conducted in the lab environment or remotely. The former is proved to be a reliable, yet expensive approach. The latter, in general, allows evaluators to focus on the analysis of the results, reduce the costs and time and enables the users to perform a test in their natural environment [26].

Remote usability testing methods may be divided into two categories [16]:

- synchronous - the evaluator can receive and manage the testing data at the same time the user performs the test, and they can hold a real-time conversation; can be done via video conferencing systems or remote application sharing tools
- asynchronous - the evaluator has no access to the data in real time and there is no interaction between them and the tester; can be done via interactive programs, questionnaires or automatic gathering data related to the user's behaviour using the system

Remote asynchronous testing allows the participants to perform the test wherever and whenever they wish. It is less intrusive for the tester than any other method and also lets them remain fully anonymous, which may encourage them to provide utterly honest answers that they might not want to give speaking to the evaluator. Furthermore, this method allows to test the system in various environments (different operating systems on different types of computers) in which the real users will be likely to use it. Taking these advantages into account, this method was chosen to evaluate the usability of the application.

5.5.1 Setup

Although conducting remote testing is quicker and easier than testing in person, time spent on setting up such a test, which is often underestimated, needs to be accounted. For the purpose of this project, a temporary code for the User Testing form was implemented. It works in the same way as the Feedback page, of which implementation is described in Section 4.7. `UserTestingComponent` provides an HTML template and a `FormGroup` object which stores the user's input. On submission, the form's content in JSON format is passed to `sendUserTestingResponse()` method in `ContactService` (see Section 4.7.1). The function stringifies the form and sends it in a body of a POST request to `'/api/user-testing/'` on the server.

The server implements a routing function which handles such a request. It extracts all the fields from the form and appropriately populates the database. A separate table was created for each task of the test and personal information. Building the User Testing questionnaire into the application makes it easily accessible for anyone using the website with no need to provide a potential tester with an external link. It makes the User Testing form inseparable part of the application, thus, any future user could take part in the usability test, which may lead to continuous improvements of the system. Additionally, it keeps all the application-related content in one place making the project more organised.

5.5.2 Questionnaire

At the top of the User Testing page, users are informed that the participation is entirely voluntary so they can attempt any questions and quit at any time they want. They are also notified that their responses are stored anonymously.

The whole test is split into seven sections; one for the user's personal information related to their experience with the Karnaugh map, five for various tasks involving using the application and one for the user's general opinion on the tool. Most of the questions ask how the user feel about the statement having performed a requested task. These questions use a five-level Likert scale where *1* means *strongly disagree* and *5 strongly agree*. The user is also given a chance to write custom comments on every task and the application.

Unfortunately, due to a small fault in the code, responses for one of the tasks (Task 3) were not saved in the database, thus, they could not be analysed. This is now fixed, so it can be used in future. The whole questionnaire can be found in Appendix B.1.

5.5.3 Results

It needs to be noted that the test is fully anonymous, therefore, its results cannot be treated as entirely reliable. However, any comments and frequent selection of the same grade for a particular question may imply there is an issue in the application that is worth investigating. As of 9 April 2018, 25 non-empty responses were received. A detailed breakdown of responses for every Likert-type question can be found in Appendix B.2.

5.5.3.1 Previous Experience

5 respondents did not specify anything about their experience with the Karnaugh map nor whether they were students. Out of the rest 20 respondents, 15 identified themselves as students and 5 as non-students. Only 5 took INF1-CL course, yet 12 heard about the Karnaugh map and 10 already knew how to use it. The low number of people who have taken the INF1-CL course may be caused by the fact the usability test was sent to a diverse group of people.

5.5.3.2 Task 0

Task 0 concerned the Tutorial page (see Section 3.1.2) and it was suggested to people who were not familiar with the Karnaugh map or struggled with some question while performing other tasks. As many as 24 respondents attempted this task. Given 10 people claimed they were already familiar with the Karnaugh map model, one may presume some of them checked the Tutorial page out of curiosity or to give feedback anyway.

All respondents agreed the information on the page was presented in a pleasant and concise way as there was no mark for this part lower than 4, leading to an average of 4.58.

Regarding the helpfulness, the average equalled 4.38. Responses consisted of mostly 5's and 4's, however, three users gave it a mark 3. Two of these respondents have never

had any experience with the Karnaugh map and one did not specify it. One of them also left a comment saying *"I didn't have any previous knowledge about Karnaugh Maps and it was tough to understand the concept without accessing external websites"*. On the other hand, 5 comments described the Tutorial as a good foundation or refreshment of knowledge about the Karnaugh map.

The Tutorial page was not a main part of the project and it was only implemented as an additional feature. Students using *Karnaugh mAPP* should have already gained background knowledge on the Karnaugh map in the INF1-CL course so it is not aimed at people who have never heard about this model. Given that, the responses are positive and nothing needs to be added.

5.5.3.3 Task 1

In Task 1 users were asked to complete any question of a type *Minimise the Expression* (see Section 3.2.6) in the Practice section.

Firstly, it asked how easy they found to locate a question. This question was attempted by 24 respondents and received only 5's and one 4 and 3, resulting in an average of 4.88. A couple of answers even included comments explicitly praising the user experience.

The second part of this task was to assess the informativeness of the feedback provided the user made any mistake in the question. As expected, the number of users who attempted this question is lower, namely 17 users. The answers were quite divergent, which might depend on in which part of the question the user made a mistake.

One user gave it 2 leaving a comment *"Couldn't understand the question and the feedback just said "wrong, think again", but I don't study informatics so maybe inapplicable to me"*. Most likely, they meant a part of initial marking appropriate states as this is the only task with a possible feedback of this kind. Feedback for marking the Karnaugh map based on some expression was described in Section 3.2.2.3. The prompt to give a question a second thought is displayed only if the user marked some states belonging to the correct solution and some states that do not. In such a case, it is difficult to give them a hint without telling the answer straight away. Moreover, the person who left that comment specified they had not taken the INF1-CL course and had no previous experience with the Karnaugh map.

Other responses included three 3's, eight 4's and five 5's. An interesting comment was *"The highlighting of which grouped state has the wrong expression is useful, however I'm not sure if it should tell you which A/B/C/D inputs are correct, as it can encourage trial and error with changing the values, rather than looking at the karnaugh map and determining the answer"*. This point was already considered during the development. It was decided to keep colouring of unit variables since it points the user to what has to be fixed rather than gives them the exact answer.

Various, yet mostly positive, responses regarding the feedback may imply it is now implemented well, however, it could be potentially improved.

5.5.3.4 Task 2

Task 2 asked the user to attempt any question of each type in the Quiz section and then rate how clear the instructions were for each of them. They were not required to complete the questions.

The first question, regarding *Label Squares* exercise (see Section 3.2.1), was answered by 24 respondents whereas all the others by 23 of them. The highest rate was obtained by *Find the Best Groups* (see Section 3.2.3), with an average of 4.65. *Name the Group* (see Section 3.2.4) received the lowest average, namely 4.48. All other questions achieved an average of above 4.5 for their instructions.

The instruction for *Name the Group* was then investigated. It used to say: ”Choose the right values for all variables for which the highlighted states evaluate to true and the other states are excluded”. Indeed, the question was imprecise, which was not noticed before. Consequently, it was changed to: ”Choose the right values for all variables whose conjunction is satisfied by the highlighted states and excluded by the others”.

5.5.3.5 Task 3

Task 3 concerned the clearness of the awarding system and how easy it is for the user to find the information about their progress in Quiz and reset this information. As already mentioned, the responses for this task turned out not to be saved properly in the database, hence, they could not be analysed. Nevertheless, the awarding system is not a crucial feature of the application, so the lack of this information should not affect the general evaluation. Moreover, the problem was already fixed so any responses regarding this task in future will be able to be examined.

5.5.3.6 Task 4

Task 4 was the last section that gave the user a certain scenario to perform. Its purpose was to assess the quality of the Instant Solver (described in Section 3.1.5). Users were prompted to find a conjunctive normal form of the expression $A \wedge B \iff (\neg C \implies D)$.

The task was performed by 21 respondents who assessed the Instant Solver fairly positively. The statement that received the lowest number of points was about the navigation. Namely, it asked how clear it was where to navigate to find the solution for the problem. The responses consisted of seven 5’s, eleven 4’s and one 3, resulting in an average of 4.38. The user who marked ”navigation” as 3, left a comment saying *”Everything was clear from the top to bottom. The only problem was that I didn’t initially know where to navigate. It would be better to put ’go to instant solver’ in the first point”*. This may suggest that the dominance of 4’s over 5’s and an occurrence of 3 might have been caused by the fact it was the first and only task that did not explicitly indicate to which tab the user should proceed.

All other aspects of this task received an average of above 4.5. Users claimed that the instructions on how to type in the expression as an input (avg. 4.67) and how to adjust

the parameters, such as switching between DNF and CNF (avg. 4.57), were clear. They also reckoned the solution was presented in a clear way (avg. 4.67) and that it was easy to see how this solution was obtained (avg. 4.62).

One of the negative comments, written by the user who did not take the INF1-CL course, said *"I can't find explained difference between cnf and dnf anywhere on this page"*. Nevertheless, as said before, this website is not supposed to teach students Logic from scratch, but serve as a supplement to the INF1-CL material. Despite that, one could consider including explicit information on DNF and CNF in the Tutorial page.

5.5.3.7 General Feedback

The last section asked to rate various properties of the application in general and asked about specific things the user thought was interesting or could be improved.

There were 22 respondents who answered the questions in this section.

The first question was how much the user agreed with a statement that the website may be beneficial for people learning about the Karnaugh map. 13 users strongly agreed (rate 5) with the statement and 8 just agreed (rate 4). Only one person rated this property with 3 (equivalent to *neither disagree nor agree*), which was the lowest grade given. Such marks led to an average of 4.55. One comment supporting this high result was: *"The app managed to present a lot of knowledge in very clear and concise way. I think that if I had a chance to use this application while taking Computation and Logic, it would certainly improve my understanding of Karnaugh Maps"*.

The second evaluated property was the ease in navigation across the website. Its results were the highest in the entire testing. Namely, twenty-one respondents marked it with 5 and one with 4, resulting in an average of 4.95. This means the website is deemed really easy to navigate.

Another property involving the user experience is the aesthetics of the website. Asked whether they believed the website's design is aesthetically pleasant, seventeen users responded with 5, four with 4 and one with 3, yielding an average of 4.73. Easy navigation and high aesthetics substantially **improve the user experience**, which translates into higher **user satisfaction**.

The last Likert-type question was to rate the application in general. The application received only high scores, namely thirteen 5's and nine 4's, which gives an **average of 4.59**.

Users particularly appreciated the progress bar in Quiz and the instant solver (its ease of use, functionality and style). A few comments suggested extending the Tutorial, e.g. by an explanation of the mathematical operators used in the expressions. Nevertheless, as already mentioned multiple times, teaching basic concepts of Logic was not a part of this project.

One user noticed sporadic delayed rendering of LaTeX-style expressions across the ap-

plication. It is the issue that was already recognised during the development. Although the configuration of MathJax, responsible for rendering, specifies to hide the expression until it is fully rendered, when it is loaded dynamically, Angular displays it instantly, before it even inserts it into MathJax's rendering queue. This causes a slight delay in rendering and the appearance of the source code for a moment. The issue was eventually dropped due to its low priority in the project. More details on how LaTeX-style text is rendered may be found in Section 4.5.5.

5.5.4 Summary of Usability Testing

The vast majority of the application's scores in the usability testing were positive. Having performed requested tasks, users found the system easy and pleasant to use, which is a proof of high **user satisfaction** and **system quality**. What is even more important, they deemed it potentially beneficial for students learning the concept of the Karnaugh map. This implies a good **information** and **service quality**. Although the responses were fully anonymous and they may not be entirely reliable, any comment may lead to noticing a flaw in the software and then fixing it. Implementing the Feedback page and keeping the User Testing form inside the application, available for every user, may increase the number of data entries, which will lead to more accurate results and more faults uncovered.

5.6 System Impact

Following the information system success model, key factors influencing `Karnaugh mAPP`'s success were presented. Given these factors, one may wonder what **(net) benefits** this application provides. The purpose of `Karnaugh mAPP` is to increase students understanding of the Karnaugh map concept, hence, this impact should be particularly evaluated. According to the usability testing, users found the application potentially beneficial for people willing to learn about the Karnaugh map. Nevertheless, this opinion cannot be treated as definite since the real impact may only be measured with students taking the INF1-CL course.

5.7 Evaluation against Existing Tools

Section 1.4 listed multiple drawbacks that existing Karnaugh map related tools possess. `Karnaugh mAPP` tried to address all these problems, which led to its following features:

- User can type in any Boolean algebra expression which they wish to display in the Karnaugh map and convert to its minimal DNF/CNF,
- Solver can output the expression in both DNF and CNF,
- Solver's grouping was tested with unit tests to ensure the correctness, i.e. that the solution is indeed minimal,

- Solver outputs *all* possible solutions,
- Solver marks all the relevant states in the Karnaugh map and lists all the groups used to produce final DNF/CNF (groups may be highlighted in the interactive Karnaugh map) to show the user the working,
- Website provides not only information on the Karnaugh map, but also numerous practice questions related to it,
- Graphical interface is user-friendly, as proved by the usability testing,
- Tool follows the responsive web design making it convenient to use on the mobile devices.

Chapter 6

Further Work

A curse of the software development is that one could endlessly implement new features or enhance existing ones. This also applies to `Karnaugh mAPP`. Therefore, given the time constraints, its final version comprises properties that were deemed the most relevant and useful. Had the time allowed, the application could be expanded by various additional features. Some of them, considered the most beneficial for the user or the course lecturer, are discussed below.

6.1 Resolution-Related Questions

Currently, the application includes questions that show the user the basic concept of the Karnaugh map as well as how to use it to convert any Boolean algebra expression to its disjunctive or conjunctive normal form, but it lacks exercises which combine the Karnaugh map with other concepts of Logic. There exists the Resolution tab (see Section 3.1.6), yet it only displays a static description of the visual representation of the resolution rule in the Karnaugh map. One could extend it by interactive examples so that the user is more engaged, thus, more likely to understand this correspondence.

Consequently, resolution-related questions could be added into the set of current exercises. For instance, the user could be presented several Logic expressions, which they first would have to mark in the Karnaugh map and then infer new expressions which are entailed by the initial ones.

6.2 More Variables

The INF1-CL introduces the Karnaugh map only with four variables. In fact, when the map includes more variables, it becomes cluttered. However, with 5 or 6 variables it is still fairly comprehensible. The Karnaugh map with more variables would allow to minimise and resolute (if resolution-related part was expanded) more complex expressions, which are more frequent in the course.

6.3 Parsing XOR Statement

The current parser does not implement XOR statements. One could add this feature in a similar way to `iffParse()` and `ifThenParse()` methods described in Section 4.5.4.

6.4 Activity Tracker

As mentioned in Section 5.4, the current version of `Karnaugh mAPP` does not implement any tool for tracking users' activity. Gathering information on how many people access the website per a unit of time, how long they spend on it and how many and which questions they attempt would allow to assess whether the application is considered practical or if students are even aware of it. Moreover, if specific data on every question was collected, such as a number of people attempting it, a total number of attempts and an average number of attempts until the question is answered correctly, could indicate with what types of exercises students struggle the most, which in turn might help the lecturer and tutors take proper action.

6.5 Admin Interface

Although the existing implementation keeps creation of questions in specific exercise services, which makes it organised and easy to find and update for people not familiar with technical aspects of the application, they are still hard-coded in the software's source code and their modification would require a new deployment. A graphical interface for an administrator (potentially the course organiser) which would allow updating the exercise set dynamically would make the application more flexible. For instance, the administrator could easily add new questions of a type with which students have problems the most frequently. Furthermore, such an interface might contain information gathered with the activity tracker described in the previous question. In general, the graphical interface would let people with low technical skills maintain and use the software as developers.

Chapter 7

Conclusion

The project's goal was to create an interactive learning tool for the Informatics 1 - Computation and Logic course at the University of Edinburgh. Out of many topics related to propositional logic, the concept of the Karnaugh map was selected since it is a powerful tool for minimisation and conversion of complex Boolean algebra expressions, which may be useful not only for Computer Science students, but also for future Electrical Engineers. The Karnaugh map model may also be used as a smooth transition to the topic about the entailment, particularly the resolution rule.

A web-based application, called `Karnaugh mAPP`, was created. It offers a large set of questions of various types in two modes. The former, Practice, allows the user to attempt any question in an uncontrolled manner. If the question is answered wrong, the user is presented feedback, usually involving colouring, to guide them to the correct answer. The latter mode, Quiz, offers a similar set of questions, yet there is no feedback other than whether the question was answered correctly or not. Instead, for every right answer, the user is awarded points which contribute to their total score, continuously displayed in the progress bar. Points are then exchanged into badges and the goal of Quiz is to collect all of them. Such elements of gamification were introduced to motivate the user to use the application. Moreover, points, working as the positive reinforcement, and practice questions, employing the observational learning, facilitate a process of learning.

`Karnaugh mAPP` was developed with Angular, which is a TypeScript framework for front-end web applications. Using Angular's powerful components, the website could be split into a series of smaller pieces with their own separate logic. It helps to keep an organised structure of the code, which itself becomes more maintainable.

Following Angular's principle, the whole application is run on the client-side. However, Feedback and User Testing forms require direct communication with the server on which the application is running. The server uses Node.js so that it can execute JavaScript code. When it receives a POST request with the form's content, it inserts the values into the database, implemented using PostgreSQL. On the other hand, points in Quiz are stored in the client's browser with the `localStorage` property.

A few unit tests were developed to ensure the correct functionality of crucial methods.

The final version of the application was also evaluated against the desired features designed based on various types of learning and a gamification technique. To get an unbiased assessment of the application and estimate the user satisfaction, the usability testing was employed. It was conducted remotely and asynchronously via an on-line questionnaire included in the application. The responses were predominantly positive. The average of general rating of the application equalled 4.59 consisting only of 4's and 5's (on a 5-level scale).

As with any software, there are still some features which potentially could be added, e.g. resolution-related questions or a graphical admin interface. These may be foundations for future honours projects.

Appendix A

Code Excerpts

A.1 Unit Test of BestGroupSolver.findBestGroups()

```
describe('Best Group Solver', () => {
    it('should find the best groups - complex', () => {
    let marked = [
      [1, 1, 1, 1],
      [1, 0, 1, 1],
      [1, 1, 1, 1],
      [1, 1, 0, 0]
    ];

    let expected1 = [
      new ExpressionGroup(false, false, null, null),
      new ExpressionGroup(null, null, false, false),
      new ExpressionGroup(null, true, true, null),
      new ExpressionGroup(true, null, false, null)
    ];

    let expected2 = [
      new ExpressionGroup(false, false, null, null),
      new ExpressionGroup(false, null, null, false),
      new ExpressionGroup(null, true, true, null),
      new ExpressionGroup(true, null, false, null)
    ];

    let expected3 = [
      new ExpressionGroup(false, false, null, null),
      new ExpressionGroup(null, true, true, null),
      new ExpressionGroup(null, true, null, false),
      new ExpressionGroup(true, null, false, null)
    ];
```

```
    let expected4 = [
      new ExpressionGroup(true, true, null, null),
      new ExpressionGroup(false, null, true, null),
      new ExpressionGroup(null, true, null, false),
      new ExpressionGroup(null, false, false, null)
    ];

    let expected5 = [
      new ExpressionGroup(true, true, null, null),
      new ExpressionGroup(null, null, false, false),
      new ExpressionGroup(false, null, true, null),
      new ExpressionGroup(null, false, false, null)
    ];

    let expected6 = [
      new ExpressionGroup(true, true, null, null),
      new ExpressionGroup(false, null, true, null),
      new ExpressionGroup(false, null, null, false),
      new ExpressionGroup(null, false, false, null)
    ];

    const result = BestGroupsSolver.findBestGroups(marked);
    expect(result).toContain(expected1);
    expect(result).toContain(expected2);
    expect(result).toContain(expected3);
    expect(result).toContain(expected4);
    expect(result).toContain(expected5);
    expect(result).toContain(expected6);
    expect(result.length).toEqual(6);
  });
});
```

Appendix B

Usability Testing

B.1 Questionnaire

1. **Personal Information**
   (a) Are you a student?
   (b) Have you taken Informatics 1 - Computation and Logic course at the University of Edinburgh?
   (c) Have you heard about the Karnaugh map before?
   (d) Do you know how to use the Karnaugh map?
2. **Task 0:** If you are not familiar with the Karnaugh map at all or you are stuck in any question because of the lack of complete knowledge about the Karnaugh map, please refer to the Tutorial page and answer the following questions:
   (a) Information on the Tutorial page is presented in a pleasant and concise way.
   (b) I found the information on the Tutorial page helpful.
   (c) Comment.
3. **Task 1:** Go to Practice tab and complete any question from Minimise the Expression set. Rate how you feel about the following statements relating to your experience:
   (a) I found it easy to find the question.
   (b) When I answered the question wrong, feedback was informative enough to find a mistake I made. (Do not choose anything if you completed the question successfully at the first attempt).
   (c) Comment.
4. **Task2:** Go to Quiz tab and attempt (no need to complete) any question of each type. Rate how clear the instructions are.

(a) Label Squares

(b) Expression to Karnaugh Map

(c) Find the Best Groups

(d) Name the Group

(e) Karnaugh Map to Expression

(f) Minimise the Expression

5. **Task 3:** Complete any question in the Quiz section. Then attempt to find the information on your general progress and clear all this information. Rate how you feel about the statements below based on your experience:

   (a) I was clearly notified whether my answer was correct and a number of points I was awarded for it.

   (b) I found it easy to find all the information about my general progress in Quiz.

   (c) I found it easy to reset all the information about my progress.

   (d) Comment

6. **Task 4:** Find the minimal conjunctive normal form of the expression: $A \wedge B \iff (\neg C \implies D)$ using `Karnaugh mAPP`. Rate how you feel about the following statements based on this experience:

   (a) It was clear where to navigate to find the solution for the problem.

   (b) The instructions on how to type in the expression were clear.

   (c) It was clear how to adjust parameters of the tool to convert the expression to the desired form.

   (d) Final solution was presented in a clear way.

   (e) It is easy to see how the final solution was obtained.

   (f) Comment.

7. **General Feedback**

   (a) I believe the website's content may be beneficial for people learning abut Karnaugh map.

   (b) I believe the website is easy to navigate around.

   (c) I believe the website's design is aesthetically pleasant.

   (d) Rate the app in general.

   (e) Comment on anything you found particularly useful/interesting.

   (f) Comment on anything you think could be improved.

B.2 Results

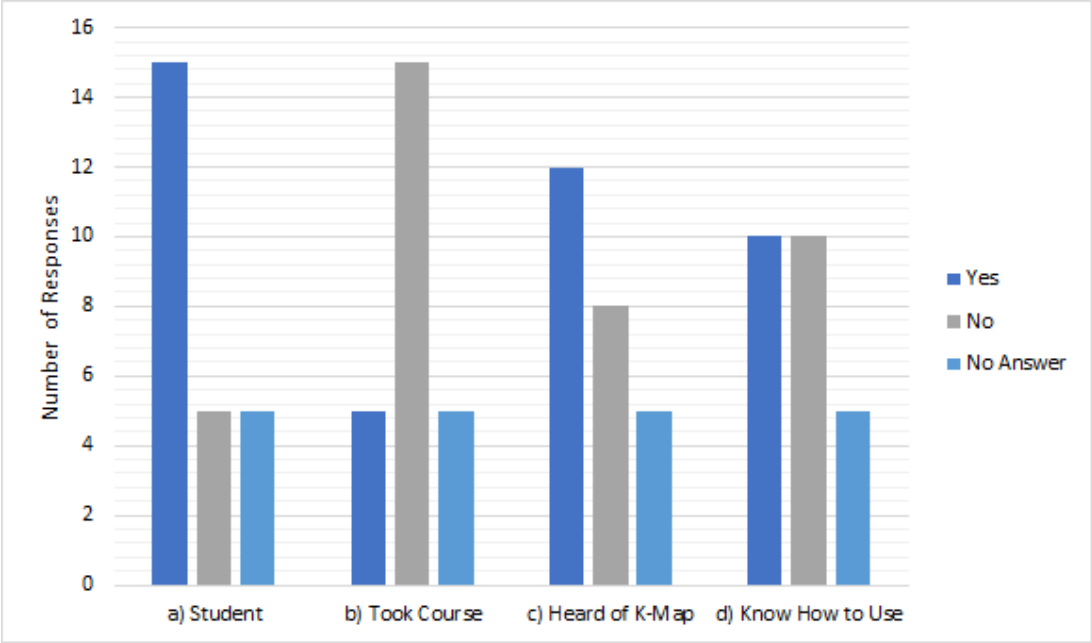

Figure B.1: Responses for Section 1 (User's Background).

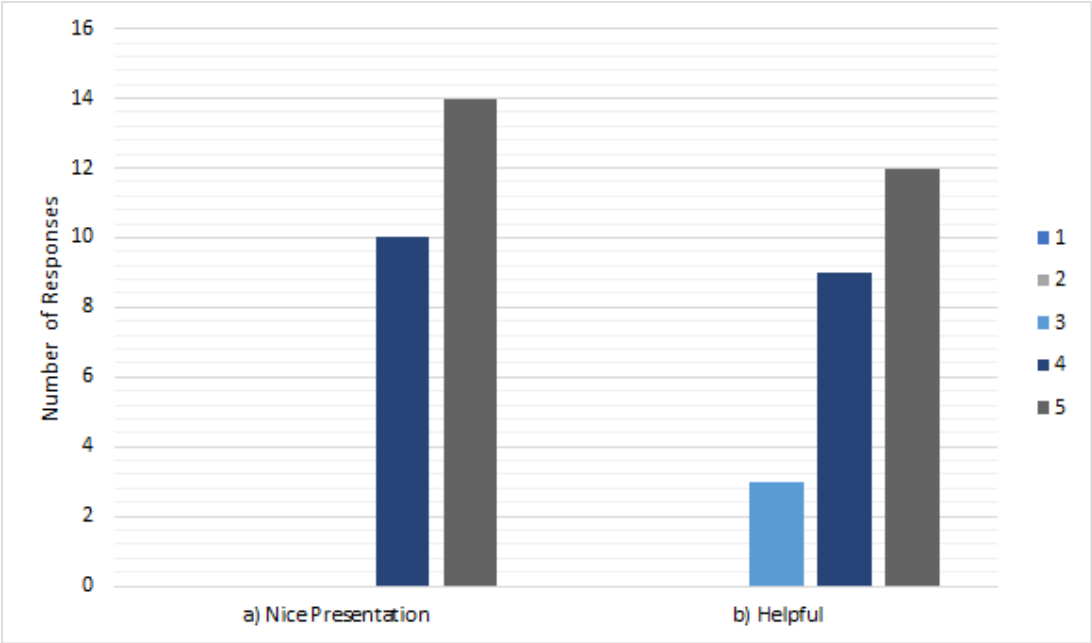

Figure B.2: Responses for Section 2 (Task 0).

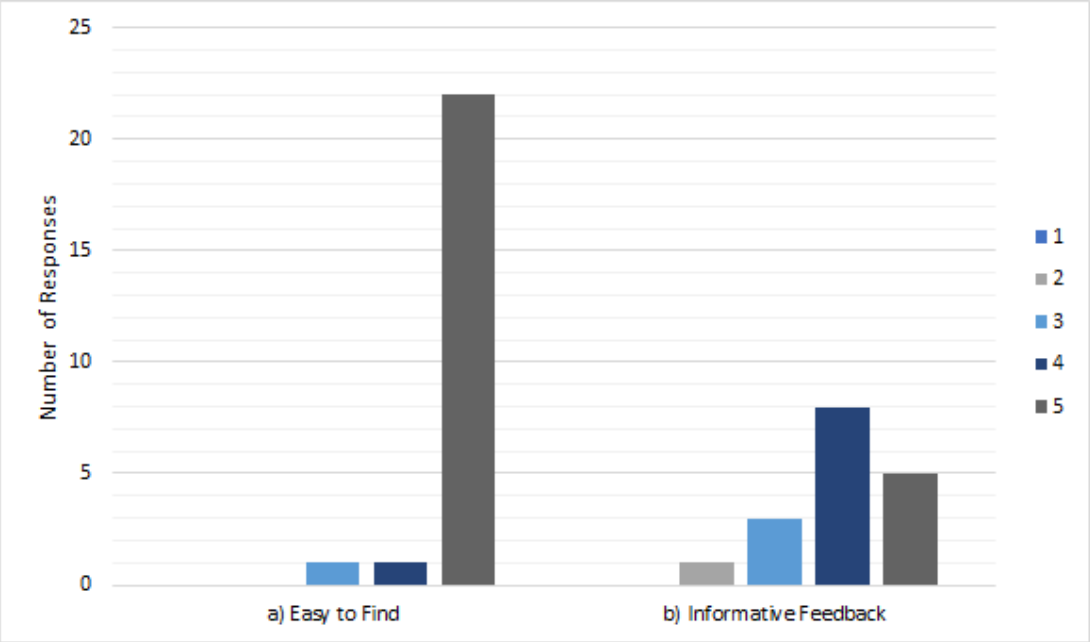

Figure B.3: Responses for Section 3 (Task 1).

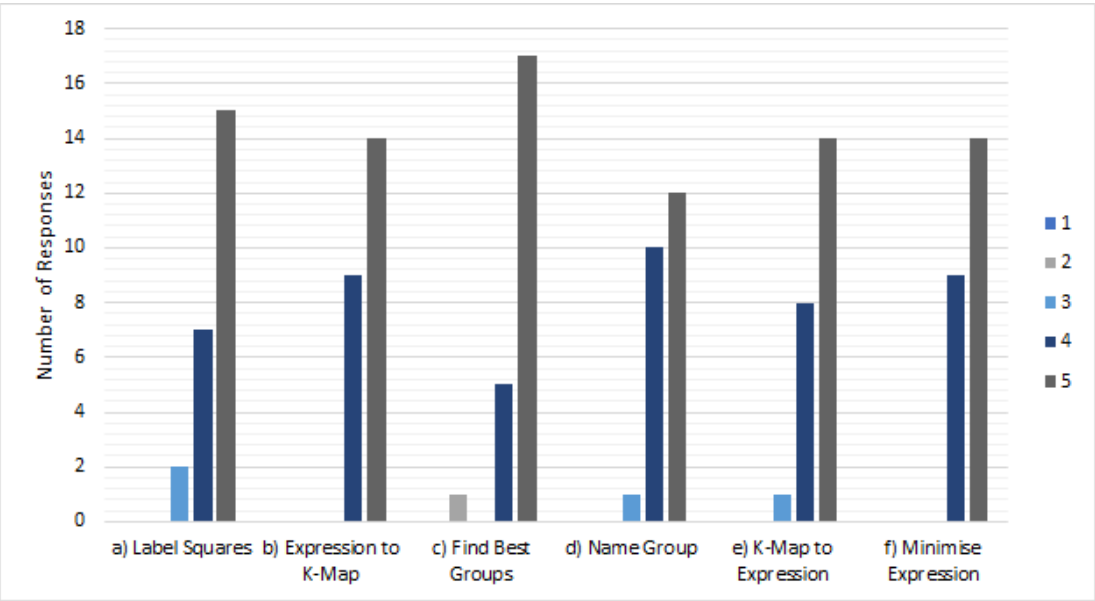

Figure B.4: Responses for Section 4 (Task 2).

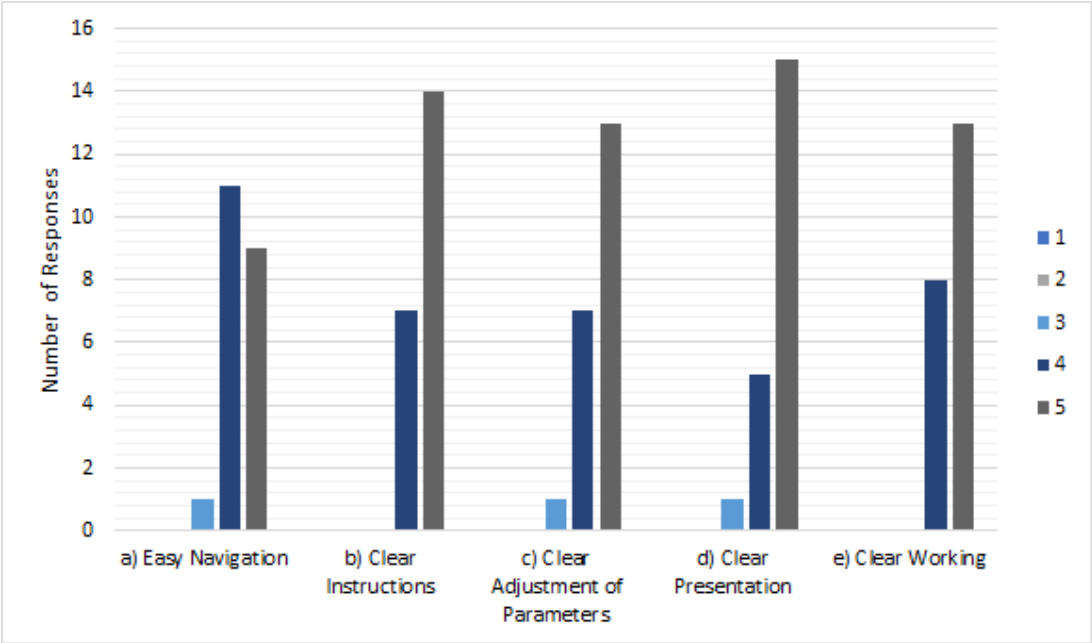

Figure B.5: Responses for Section 6 (Task 4).

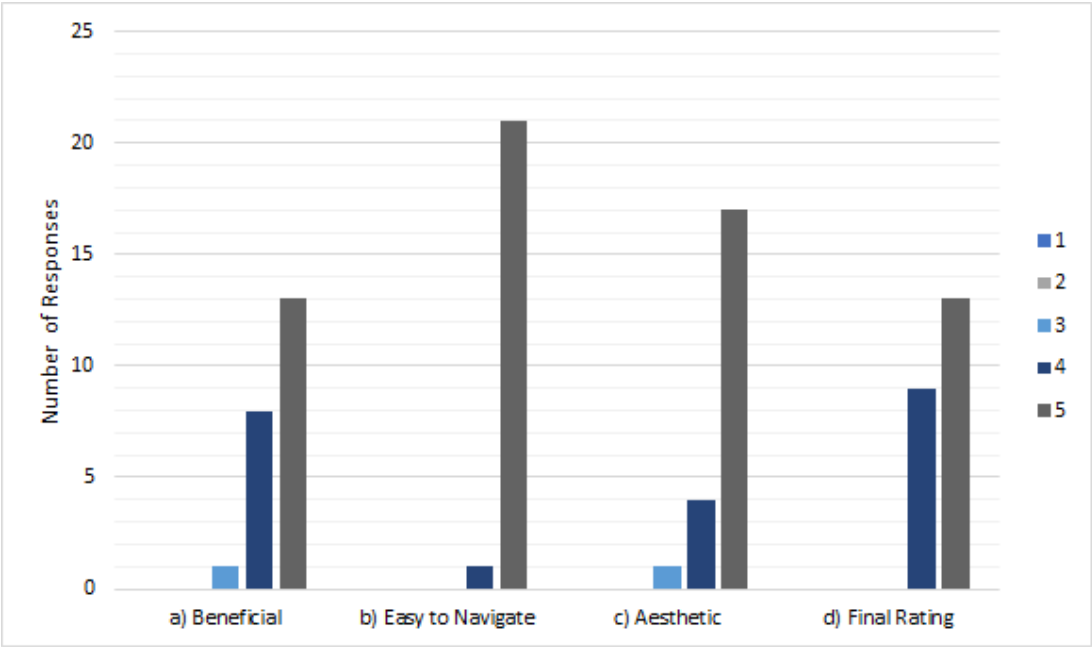

Figure B.6: Responses for Section 7 (General Feedback).

Bibliography

[1] Karnaugh Map Explorer 2.0. http://www.ee.calpoly.edu/media/uploads/resources/KarnaughExplorer_1.html. Accessed on 26 February 2018.

[2] Karnaugh-Veitch Map. http://www.mathematik.uni-marburg.de/~thormae/lectures/ti1/code/karnaughmap. Accessed on 26 February 2018.

[3] Mathematical expression evaluator in JavaScript. https://github.com/silentmatt/expr-eval. Accessed on 17 January 2018.

[4] MathJax | Beautiful math in all browsers. https://www.mathjax.org. Accessed on 18 January 2018.

[5] node-postgresql. https://node-postgres.com. Accessed on 12 March 2018.

[6] ReactiveX Documentation: Observable. http://reactivex.io/documentation/observable.html. Accessed on 12 March 2018.

[7] Logic circuit simplification (sop and pos). http://www.32x8.com, 2014. Accessed on 26 February 2018.

[8] *Introduction to Psychology*, chapter Learning, pages 266–300. University of Minnesota Libraries Publishing, Minneapolis, MN, 2015. Adapted from a work produced and distributed under a Creative Commons license (CC BY-NC-SA) in 2010 by a publisher who has requested that they and the original author not receive attribution.

[9] PostgreSQL: About. https://www.postgresql.org/about, 2018. Accessed on 12 March 2018.

[10] A. Butterfield and G. E. Ngondi. information system. In *A Dictionary of Computer Science*. Oxford University Press, 7th edition, 2016.

[11] C. Commons. Attribution-NonCommercial 4.0 International (CC BY-NC 4.0). https://creativecommons.org/licenses/by-nc/4.0. Accessed on 4 April 2018.

[12] E. L. Deci, R. Koestner, and R. M. Ryan. A meta-analytic review of experiments examining the effects of extrinsic rewards on intrinsic motivation. *Psychologial Bulletin*, 125(6):627–668.

[13] W. H. DeLone and E. R. McLean. The DeLone and McLean Model of Information Systems Success: A Ten-Year Update. *Journal of Management Information Systems*, 19(4):9–30, 2003.

[14] S. Deterding, D. Dixon, R. Khaled, and L. Nacke. From Game Design Elements to Gamefulness: Defining "Gamification". In *Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments*, pages 9–15, New York, NY, USA, 2011. ACM.

[15] D. Dicheva, C. Dichev, G. Agre, and G. Angelova. Gamification in Education: A Systematic Mapping Study. *Educational Technology Society*, 18(3):75–88, 2015.

[16] S. Dray and D. Siegel. Remote Possibilities?: International Usability Testing at a Distance. *interactions*, 11(2):10–17, March + April 2004.

[17] Emant Pte Ltd. Karnaugh Map Calculator. `http://electronics-course.com/karnaugh-map`. Accessed on 26 February 2018.

[18] European Parliament and European Council. Directive 2009/136/EC. `http://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32009L0136`, 2009.

[19] S. Fenton. *Pro TypeScipt: Application-Scale JavaScript Development*, chapter TypeScript Language Features, pages 1–62. Apress Media, Basingstoke, United Kingdom, 2nd edition, 2018.

[20] M. Fourman. Informatics 1 - Computation Logic: Tutorial 2 - Propositional Logic: Karnaugh Maps. `https://www.inf.ed.ac.uk/teaching/courses/inf1/cl/tutorials/2017/solutions4.pdf`, 2017. Accessed on 12 October 2017.

[21] A. Francisco-Aparicio, F. L. Gutiérrez-Vela, J. L. Isla-Montes, and J. L. G. Sanchez. *Gamification: Analysis and Application*, pages 113–126. Springer London, London, 2013.

[22] Google LLC. Angular Docs. `https://angular.io/docs`, 2018. Accessed on 16 March 2018.

[23] J. Hamari, J. Koivisto, and H. Sarsa. Does Gamification Work? - A Literature Review of Empirical Studies on Gamification. pages 3025–3034. IEEE, January 2014.

[24] M. Karnaugh. The Map Method for Synthesis of Combinational Logic Circuits. *Transactions of the American Institute of Electrical Engineers, Part I: Communication and Electronics*, 72(5):593–599, Nov 1953.

[25] T. R. Kuphaldt. *Lessons in Electric Circuits vol. IV*, chapter 8: Karnaugh Mapping. 2015.

[26] R. Martin, M. Al Shamari, M. Seliaman, and P. Mayhew. Remote Asynchronous Testing: A Cost-Effective Alternative for Website Usability Evaluation. 03:99–104, January 2014.

[27] A. Matallaoui, N. Hanner, and R. Zarnekow. *Introduction to Gamification: Foundation and Underlying Theories*, pages 3–18. Springer International Publishing, Cham, 2017.

[28] J. McDonigal. *Reality Is Broken: Why Games Make Us Better and How They Can Change the World*. Penguin, New York, 2011.

[29] E. D. Mekler, F. Brühlmann, A. N. Tuch, and K. Opwis. Towards understanding the effects of individual gamification elements on intrinsic motivation and performance. *Computers in Human Behavior*, 71:525 – 534, 2017.

[30] Mozilla. Web technology for developers. `https://developer.mozilla.org/en-US/docs/Web`. Accessed on 20 February 2018.

[31] N. Murray, F. Coury, A. Lerner, and C. Taborda. *ng-book: The Complete Book on Angular 4*. Fullstack.io, San Francisco, CA, 2017.

[32] D. L. Nelson, J. C. Quick, A. Armstrong, and J. Condi. *ORGB*, chapter Learning and Performance Management, pages 91–103. Cengage Learning Inc., 2nd edition, 2007.

[33] S. Nicholson. A User-Centered Theoretical Framework for Meaningful Gamification. In *Games + Learning + Society*, volume 8, pages 223–230, June 2012.

[34] Node.js Foundation. Express Guide: Using middleware. `https://expressjs.com/en/guide/using-middleware.html`, 2017. Accessed on 12 March 2018.

[35] Pivotal Software, Inc. Jasmine Documentation. `https://jasmine.github.io/`, 2018. Accessed on 16 March 2018.

[36] A. K. Przybylski, C. S. Rigby, and R. M. Ryan. A motivational model of video game engagement. 14(2):154–166, 2010.

[37] R. M. Ryan and E. L. Deci. Intrinsic and Extrinsic Motivations: Classic Definitions and New Directions. *Contemporary Educational Psychology*, 25:54–67, 2000.

[38] E. L. Thorndike. *Animal Intelligence: Experimental Studies*, chapter Laws and Hypothesis for Behaviour, pages 241–281. The Macmillan Company, New York, 1911.

[39] C. H.-H. Tsay, A. Kofinas, and J. Luo. Enhancing student learning experience with technology-mediated gamification: An empirical study. *Computers Education*, 121:1–17, 2018.

[40] E. W. Veitch. A Chart Method for Simplifying Truth Functions. In *Proceedings of the 1952 ACM National Meeting (Pittsburgh)*, ACM '52, pages 127–133, New York, NY, USA, 1952. ACM.

[41] J. Zagal, M. Mateas, C. Fernández-Vara, B. Hochhalter, and N. Lichti. Towards an Ontological Language for Game Analysis. In *Proceedings of International DiGRA Conference: Changing Views - Worlds in Play*, pages 3–14, January 2005.

[42] F. Ziegelmayer. Karma - How It works. https://karma-runner.github.io/2.0/intro/how-it-works.html. Accessed on 16 March 2018.