Dynaslides

Francisco José Valente Gomes Campaniço

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisor: Prof. Daniel Jorge Viegas Gonçalves

Examination Committee

January 2021

Acknowledgments

First and foremost, I would like to thank Professor Daniel Gonçalves for giving me the opportunity and always being available to help in this challenging project. Without his guidance, I would not be able to achieve the results from the development of Dynaslides.

Additionally, I would like to thank my family, especially my mother and father, for providing me with a step forward in my education and ultimately giving me more opportunities in the future.

. Furthermore, I would like to thank my friend and colleague João Rafael, for providing me with additional knowledge and wisdom, which helped me get through all the process of making this objective in my education.

Finally, I would like to thank my personal group of friends called ”Algang” specifically, Lucas, Pedro, Manuel, Hugo and Matilde, and Paulo and Rodrigo for their support and companionship through the execution of this project.

Abstract

In order to display information to a relatively large audience, we often make use of multimedia presentations. However, creating these presentation documents requires the use of specific applications based on outdated presentation techniques and that do not take full advantage of contemporary technologies. Most of these tools are only meant to create documents that showcase information and which do not offer the possibility for the audience or the presenter to interact with the content during the actual presentation. Therefore, DynaSlides proposes a solution that relies on a standard, established, open-source framework to provide several interactive and dynamic features without needing to exit the presentation environment. It also features an extensible plugin structure, capable of handling additional interactive components more easily, excluding the current functionalities. Moreover, usability tests are performed on the initial prototype to make sure it meets the desired requirements.

Keywords

Dynamic; Slideware; Presentations; Powerpoint; Multimedia; Interactivity.

Resumo

Para exibir informações a um público relativamente grande é necessário o uso de apresentações multimédia. No entanto, a criação desses documentos de apresentação requer o uso de aplicações específicas baseadas em técnicas de apresentação desatualizadas e que não retiram total proveito das tecnologias contemporaneas. Muitas destas ferramentas destinam-se apenas a criar documentos que apresentem informação e não oferecem ao público ou ao apresentador a possibilidade de interagir com o conteúdo durante a própria apresentação. Portanto, a ferramenta DynaSlides oferece uma solução baseada numa estrutura padrão, estabelecida e de código aberto de forma a fornecer diversos recursos interativos e dinâmicos sem a necessidade de sair do ambiente de apresentação. A ferramenta também possui uma estrutura de plugins extensível, capaz de lidar com componentes interativos adicionais mais facilmente, excluindo as funcionalidades atuais. Para além disso, testes de usabilidade são realizados no protótipo inicial para verificar se a ferramenta corresponde aos requisitos desejados.

Palavras Chave

Dinâmico; Apresentações; Powerpoint; Multimídia; Interatividade.

Contents

List of Figures

1 Introduction

Contents

Sharing knowledge and communicating ideas is part of what makes us human. From narrating a simple tale to describing a solution for a complex problem, our society thrives when information is analyzed, distributed, and stored for future generations. There are many different systems created to help diffuse and preserve this valuable information. There is a contemporary technique, in particular, we are going to analyze carefully, often referred to as a multimedia presentation.

This method of conveying information is usually associated with a linear sequence of different slides (sections), each one representing a fraction of the idea the creator wants to deliver. It requires customized software to generate presentations, allowing different representations of information. Nowadays, however, these standard presentation applications do not take full advantage of the potential in presenting with access to several different technologies. Many solutions show similarities to physical slides. Each one has some text, possibly accompanied by a visual guide, such as an image or a video. It is a significant improvement over older presentation techniques, such as overhead projectors that require transparencies (transparent flexible sheets, typically made of cellulose acetate, where the user can draw text and figures) to display information in larger areas. Unfortunately, presentation tools are surprisingly too similar to the aforementioned outdated technique. There is no standardized application that uses the vast different interactive methods to convey information to the end-user.

Therefore, this project focuses on the field of multimedia presentation applications, how they are organized and convey information to a target audience, and understanding if that audience can correctly perceive the information that is shared.

1.1 Motivation

One major dilemma regarding this subject arises when a presenter wants to interact with an audience and requires an external application. Stopping the presentation and using a different tool to showcase an interactive feature might interrupt its flow and change how the audience perceives the information displayed. It is also inconvenient for the presenter to switch between applications since it often requires preparing before the presentation starts and dragging windows into the right view, leading to unexpected problems. We consider that nowadays, interaction with the audience becomes more and more of a necessity. Most presentation environments have access to the internet. There should be a presentation application that offers these features while being practical and intuitive.

Another glaring problem with contemporary presentation environments is the lack of interaction between the audience and the presenter. The one that exists happens via verbal communication and feedback, such as questions or commentaries. If the presentation utilizes a slideware application in a technological environment with access to the internet, there should be an easy and intuitive approach to receive feedback from the audience. Not only could the presenter offer an immediate response based on the reactions, but he could also change future presentations based on that same feedback.

The relevance for this type of application is also exacerbated by the current COVID-19 pandemic we currently face. Due to the increasingly higher usage of online lecture and presentation environments, giving oral feedback became even more difficult than before. To have an application that can gather feedback information without interrupting the flow of the presentation appears to be even in more demand than ever before.

1.2 Objective

The goal of this project is to **design a dynamic presentation tool that allows the inclusion of rich multimedia content while fostering interaction between the presenter and the audience**, and without compromising relevant traits already acquired by contemporary presentation applications, such as how difficult it is to produce a presentation. This means our developed application is a proof of concept of an interactive presentation environment that shows the possibility of increasing the interactivity of slide-based presentations without losing the possibility of using an already established presentation application.

Since keeping the presentation's flow uninterrupted is an important feature to acquire, providing several interactive features inside the presentation view implies the creator does not need to exit the presentation environment to benefit from them. For example, if a user wishes to change the representation for a given dataset or add/edit data, he should have the ability to do so while interacting with the presentation and audience.

Adding new features to an already established model usually means the difficulty of using the application will somewhat increase, thus being important to attenuate the impact of learning and ultimately utilizing new components. Dynaslides works alongside **Reveal.js** to correctly assess Microsoft's Powerpoint presentations and then offer a couple of interactive features that can be used in a more recent digital format.

1.3 Document Structure

In Section 2, we investigate state of the art in the industry of presentation software, taking into account different techniques described in several related papers, as well as different applications that take a share of the target market. Moving forward, in Section 3, using the motivation and aforementioned research, we define a set of requirements and design a solution for a dynamic presentation application. Finally, we state the results of evaluating the resulting tool in section 4, then we mention some final remarks, the limitation of our work and what can be further improved in Section 5.

All the code of this project and the respective reference guide can be found in the following git repository:

`https://github.com/Nytrum/Dynaslides-Thesis-Prototype.git`

2

Related Work

Contents

To solve the aforementioned problems, we start by finding the current practices in multimedia presentations and use them to draft a suitable solution. The following material is, therefore, categorized into three different sections to showcase similarities between different papers:

- **Navigation** - different styles of presenting and how the information reaches the audience;
- **Interactivity** - different approaches of interacting and conveying information to a target audience;
- **Authoring** - different ways of authoring documents/presentations (how to create and edit presentations) and how several existing techniques can lead to better authoring environments.

2.1 Overview

Static information tools (linear sequences of slides with little to no interaction with the respective audience) dominate the presentation application market, such as Microsoft PowerPoint [1] or Apple's Keynote [2]. They rely on standard one-way linear sequences [7] focusing more on displaying information to the end-user and associating it with a custom aesthetic. Therefore, we can extrapolate that presentations do not require a narrative or a storyline to link produced slides, often associated as an effective tool to share information with others [8]. We can also say these presentations do not allow the user to interact with the audience since there is no way to view and use interactive elements while presenting or to have several directions from one slide to another. Google Slides [3] already tries to integrate more interactive features by using a built-in component with a variety of plugins made by the community. These plugins can also be rated by the users and usually require Google Chrome, considering they work as extensions for it.

There are several key features this tool requires to achieve the desired goals of this report. We will show several attempts made by other agents, and correlate the results with the tool we are developing, explaining what can be improved or used to achieve substantial findings.

2.2 Navigation

Navigation techniques on presentation slideware have not changed must in the past. The industry standard, Microsoft PowerPoint, uses a linear sequence of slides to convey a message and is often lacking at telling the complete story since slides are envisioned one by one without requiring a plan or overview to connect them. In particular, PowerPoint has been associated with having a ”dubious reputation” [9], turning the vast number of users ”into bullet-point dandies” [10], criticized for encouraging

[1] www.microsoft.com/Microsoft/PowerPoint
[2] https://www.apple.com/keynote/
[3] https://www.google.com/slides/about/

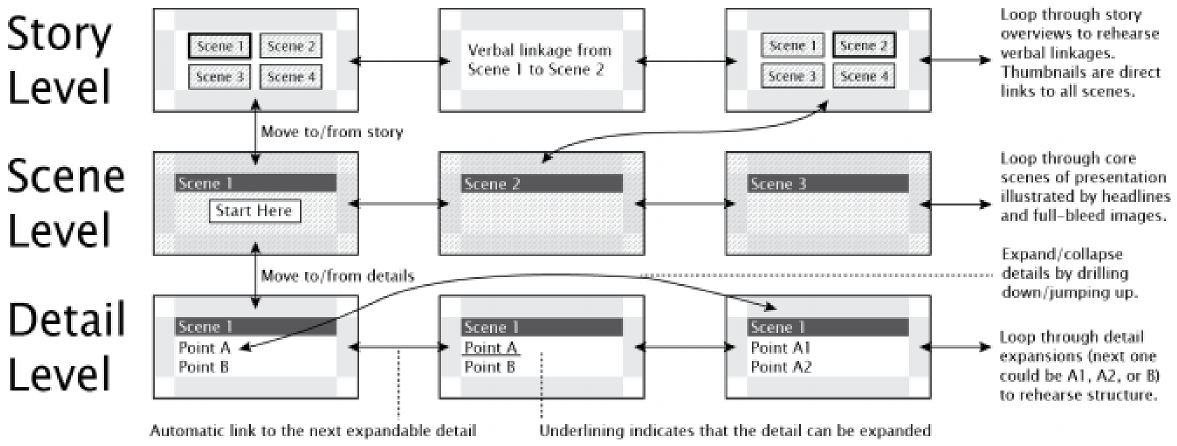

Story Level
Scene Level
Detail Level
Scene 1
Scene 2
Scene 3
Scene 4
Verbal linkage from Scene 1 to Scene 2
Loop through story overviews to rehearse verbal linkages. Thumbnails are direct links to all scenes.
Move to/from story
Start Here
Loop through core scenes of presentation illustrated by headlines and full-bleed images.
Move to/from details
Expand/collapse details by drilling down/jumping up.
Point A
Point B
Point A1
Point A2
Loop through detail expansions (next one could be A1, A2, or B) to rehearse structure.
Automatic link to the next expandable detail
Underlining indicates that the detail can be expanded

Figure 2.1: HyperSlides presentation topology [1]

oversimplification of complex information [11] [12], and described by experts as a negative contribute for ”dialogue, interaction, and thoughtful consideration of ideas” [13].

Some existing applications, such as Prezi [4], pptPlex [5] or Impress.js [6], often called Zoomable User Interfaces (ZUIs) [14], already undertake this issue by using animations and zoom capabilities as a tool to escape traditional linear navigation. However, this is only one approach for non-linear navigation and refers specifically to the visual characteristic of presentations.

HyperSlides [1], which offers another look to this problem, is a tool that started by conducting a grounded theory to understand better the idea of presenting with a slideware application and discovering the potential for a more dynamic approach to prototyping slide presentations. It acknowledges the need for this planning, referring to it as *Planning with Points*, one of the main four requirements of the application. This representation demands creating a storyline (high-level planning of so-called scenes with a respective linkage between them) to ensure the narrative is coherent throughout the entire presentation (data manipulation happens before creating their physical manifestation, to promote structuring of ideas). Moreover, it avoids a problem like procrastination, which might occur when creating slides with no clear connection behind them. Figure 2.1 shows a representation of the different levels and how slides are connected.

NextSlidePlease [2], a support presentation tool, pursues the navigation and authoring problem of standard presentation tools by creating an editing environment that supports the following features:

- **Graph-based authoring** - the tool uses a graph with weighted connections, each encoding the time estimate and the relative priority of the slide. The graph also allows the user to have a subset of paths that are optimized according to time and priority, therefore providing a more effective

[4] https://prezi.com
[5] https://www.microsoft.com/en-us/download/details.aspx?id=28558
[6] https://impress.js.org

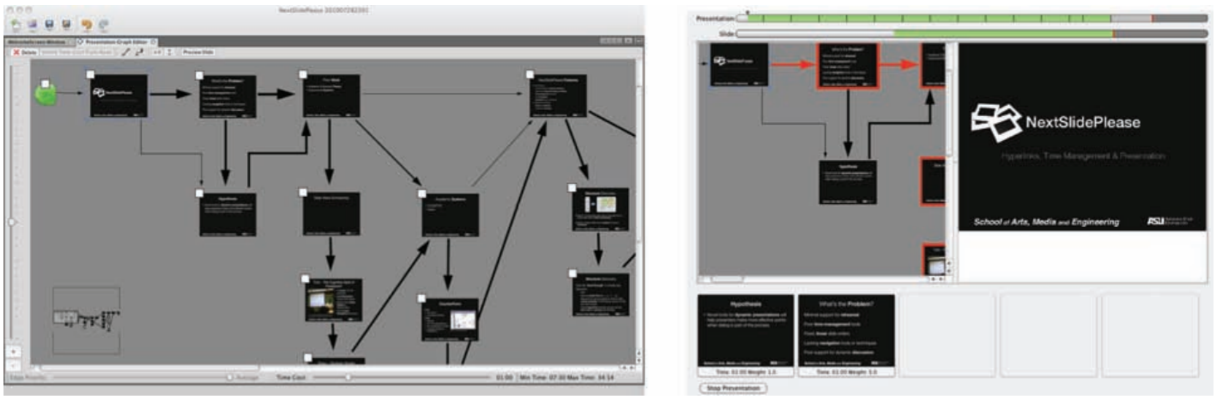


Figure 2.2: Authoring and presentation environments demonstrated in NextSlidePlease [2]

navigation experience;

- **Content reuse and Path suggestion** - the algorithms created can automatically suggest paths (based on the content of the slides) and effectively lower the authoring time;
- **Time management** - not only executed by the aforementioned points, but also by providing a user-defined interface that displays the time limit for each slide with the corresponding path suggestion. Now the user can control the pace of each slide using the time displayed in the interface.

Figure 2.2 demonstrates the four characteristics in their respective interactive environments, with the first being the most important when it comes to the navigation paradigm. The graph-like structure allows for more intuitive navigation. It solves the issues with slides not having a clear connection when separated by intermediate slides in linear navigation.

Some presentation tools like MindXpres [3], a content-driven cross-media platform with a modular architecture and a plugin mechanism, try to solve many issues in today's standard presentations such as linear navigation, separation of content and presentation, extensivity and interactivity. It uses a dedicated document format to avoid authoring in languages such as HTML or XML to resolve them. It requires a compiler, in the form of a Node.js tool with a custom XML schema, to produce a self-contained presentation bundle that couples the information gathered in the document and a portable, cross-platform presentation runtime engine for more interactive and connected presentations. All the above functionality creates the core of the application. Further development uses a system of plugins composed of components, containers, and structures while being implemented as JavaScript bundles, with the individual CSS files. For example, since the tool does not offer a default presentation environment (as not to require linear navigation), a plugin was created that mimics the standard linear model to serve as a proof of concept for the application.

Another tool that acknowledges the need for previously thought guidelines is a MindXpres interactive data plugin [8], which adds to the idea by making interactive data visualization with different states. Each

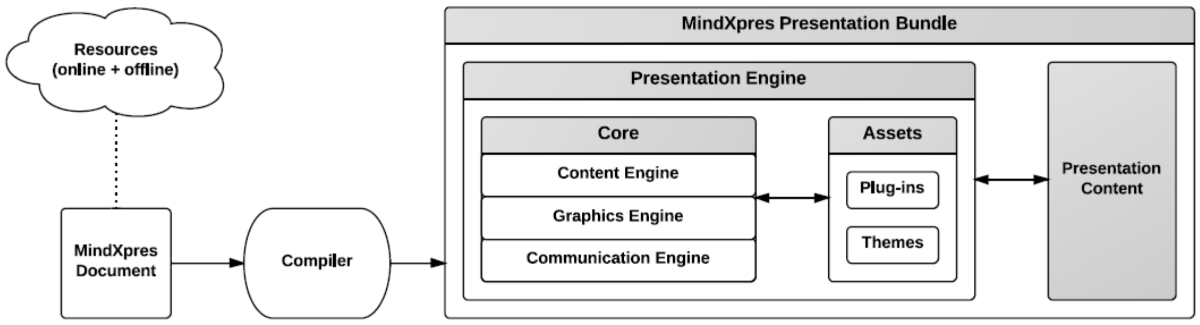


Figure 2.3: MindXpres architecture [3]

one conveys the message the author wants to pass to the audience. It uses d3.js (a javascript library, which creates data-driven visualizations) and the structure of MindXpres to implement a presentation with several states corresponding to a representation of the given data. It also offers the user a live editing menu to further interact with the previously mentioned data.

Predefining a sequence of different views or states for a given data model allows for better exploration of the data and synchronizes with the thought narrative beforehand.

Multipresenter [5] is yet another application that offers a different look into the navigation problem. It states that most slideware tools are focused on presentation content authoring and do not support the dynamic aspect of presentations. This work highlights key elements that standard applications usually do not address, such as non-linear presentation styles, revisiting earlier information (many presenters

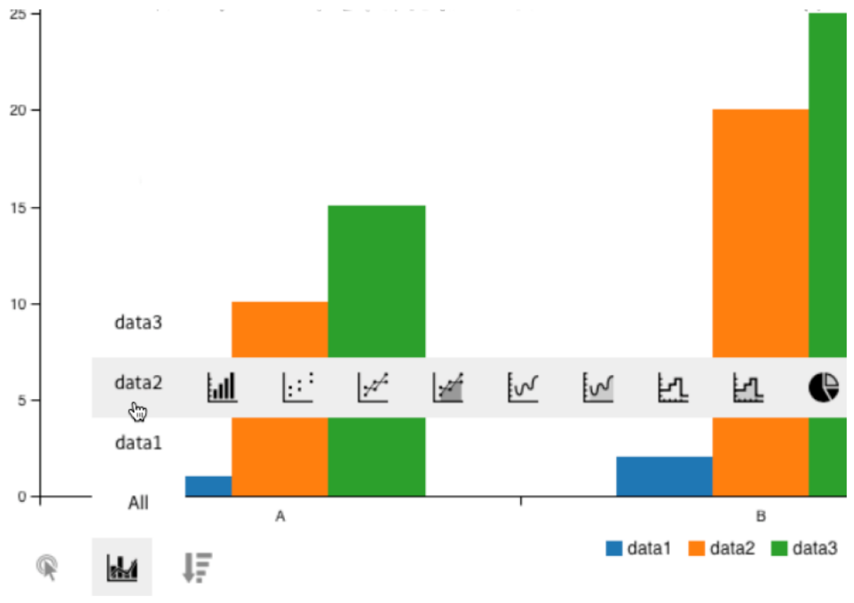


Figure 2.4: Interactive data slides menu [4]

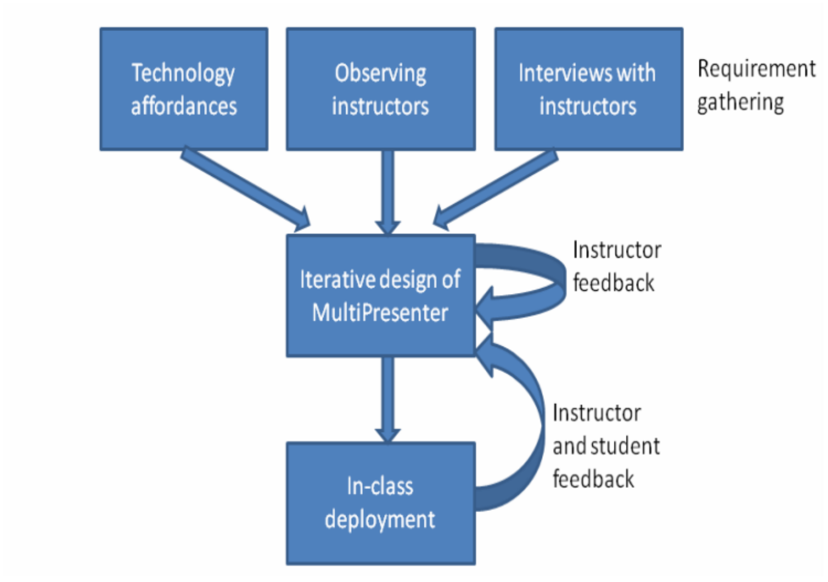


Figure 2.5: Multipresenter [5] user-driven approach

go back to ideas mentioned in previous slides, reinforcing the need for non-linear navigation), creative use of several sliding whiteboards (most classrooms have access to more than one whiteboard), mental concepts which benefit from a larger space (instructors claimed that they needed more space for complex ideas) and presentation as a dialogue. Thus, the application's main focus is to separate the content from the presentation, distinguishing between the content, layout, and presentation layers while focusing more on the presentation layer.

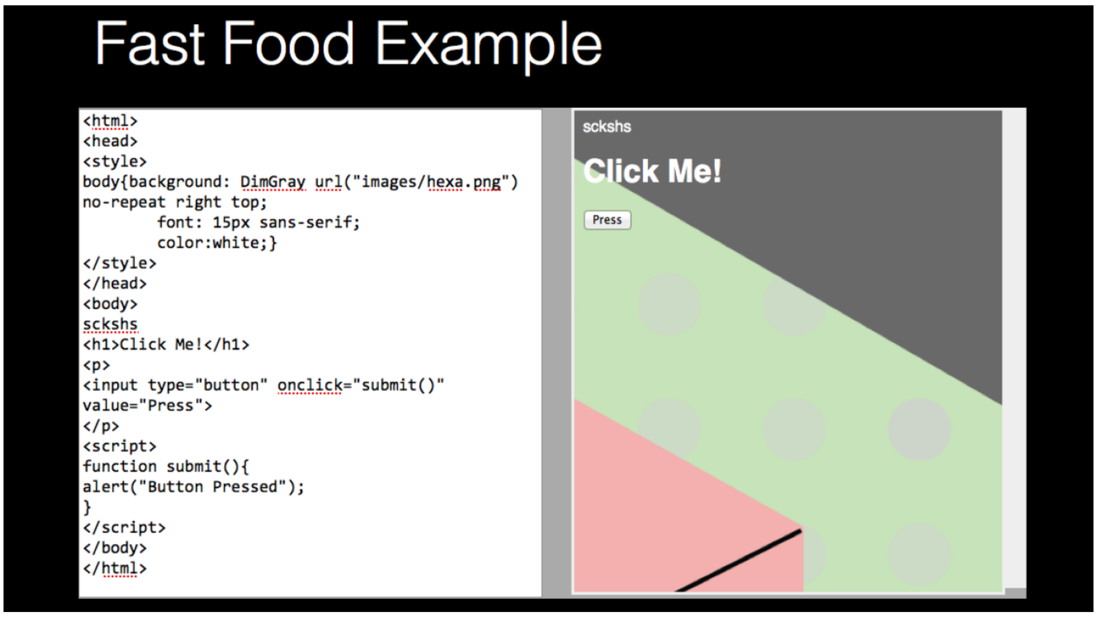
To achieve this, the application took a user-driven approach (Figure 2.5), focusing on the presenter by giving an intuitive interface and focusing on the audience by using several screens content separation.

2.3 Interactivity

Interactivity is one of the most important features in this report since the main goal is to give the end-user the ability to have real-time interaction with the audience and adjust the presentation to the audience's needs.

SlideDeck.js [6] is a multimedia presentation framework that includes a custom markup language with its respective macro processor, an interaction engine, and a course content browser.

A custom markup language ensures the user follows a strict set of rules. A macro processor can correctly generate the respective HTML tags and make the desired visualization, making the tool have a certain level of abstraction to avoid the user creating the HTML visualization. An interactive engine creates the basic interaction between slides to ensure the application works as a standard presentation


Fast Food Example

```
<html>
<head>
<style>
body{background: DimGray url("images/hexa.png")
no-repeat right top;
        font: 15px sans-serif;
        color:white;}
</style>
</head>
<body>
sckshs
<h1>Click Me!</h1>
<p>
<input type="button" onclick="submit()"
value="Press">
</p>
<script>
function submit(){
alert("Button Pressed");
}
</script>
</body>
</html>
```

Figure 2.6: SlideDeck.js code slide [6]

tool.

Finally, the course content works by displaying slide names and descriptions so that students can easily navigate through them. It also mentions the need for another type of interactivity and tries to implement it by creating interactive coding exercises. The tool features a live editing window in a slide prepared to handle it, allowing an audience member to test the code and to learn more about the material provided. The method will enhance the presentation, giving it a more dynamic and interactive style and promoting a more engaging audience.

However, this framework is not the only one to mention this need for interactive code exercises in presentations. There is an article that aims at enhancing programming lectures using interactive web-based lecture slides [15] by conducting a case study using a well-defined web presentation framework called **Reveal.js** [7]. It allows for server technology to be used, giving more options when implementing interactive content and allowing more flexibility by using web-based languages like HTML, CSS, and Javascript. The article implements server-client presentations, described by **Reveal.js** as multiplexing, which implies there are two versions of the presentation, one the creator uses and displayed for the whole class, and another one that all the attendees of the lecture can view on another device. The client-side version allows users to have a better viewing experience, improved accessibility, the ability to interact with slides (click on a button or running code), and perform tests or questions using their device. Thus, the creator has access to the instruments from standard presentation methods and other interactive features to convey the presenter's views as appropriate. The article also adds up by saying

[7] https://revealjs.com/

the study results were overall positive, with the majority of responses being ”agree” or ”strongly agree”. It further supports the claim that interactive based content on a lecture/presentation is a reliable method of conveying information for presentations that justify it.

MindXpres also supports a plugin [8] that seeks to implement interactive coding in a presentation tool. It starts by defining seven requirements for the plugin, which are as follows:

- **Automatic indentation and syntax highlighting** - All code editors already support this, and since this can be a burden for many programmers, the tool should feature it;
- **Efficient navigation of source code** - Having the ability to scroll through the code is a priority due to the slide size constraints and some programs code execution jumping back and forth;
- **Visualise the working of the code** - Visual aids can help with the production of a mental model for a section of code. Dynamic media brings measurable improvements in knowledge transfer over the use of static media [16];
- **Integration in presentation tools** - Integrating the functionality in the presentation decreases the need for changing between different applications;
- **Extensible support for multiple languages** - Having the ability to support multiples languages increases the overall versatility of a given application;
- **Extensible visualisation choices** - Different visualizations for different executions plays an import role in the construction of a mental model [17];
- **Interactive program execution** - Being able to change different inputs and lines of code augments the presentation's interactivity.

To implement these requirements, the plugin features an architecture that employs language modules to support as many languages as there are modules available for them. The language is detected automatically, and then, the language module creates a generic execution log (programming languages are converted to the JSON notation) to be used by a visualization engine (Figure 2.7). Finally, the plugin will be used in the MindXpres [3] application and can be used in the standard presentation format (Figure 2.8).

. The paper about the MindXpress interactive data plugin [8] also provides great insight into another form of interactivity, which relies on different interpretations for a given dataset. It achieves that by using a menu designed to change between different views while the presenter conveys the correlated narrative. It also allows the audience to further understand the dataset by viewing different variants and extracting as much from it as possible.

The paper also provides another feature to interact with the platform, which involves giving the presenter the possibility to change the representation or the presentation parameters, allowing for a more

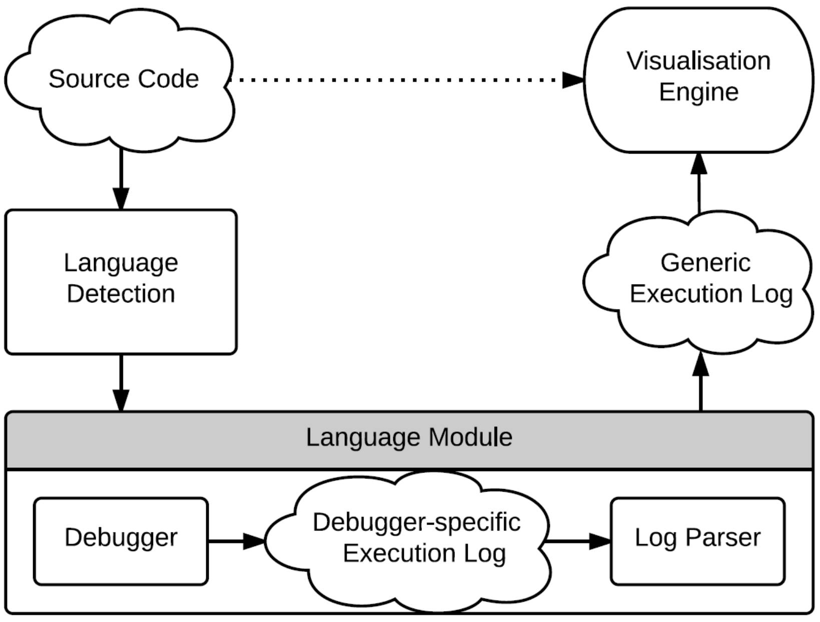
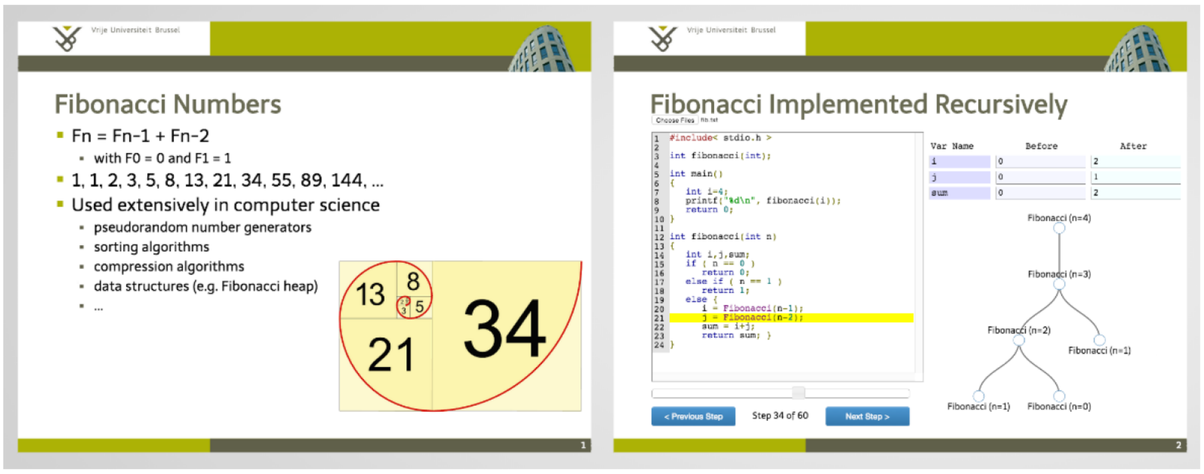


Figure 2.7: Architecture representation of the source code plugin for MindXpres [3]

audience-driven presentation. Finally, a final point in the MindXpres plugin [8] demonstrates the possibility for recording and consequential playback of the presentation since it may change during runtime.

2.4 Authoring

Fast and reliable authoring is something a creator always values in a presentation tool since it allows for better time management and an easier replication of previous work.



Fibonacci Numbers

- Fn = Fn−1 + Fn−2
  - with F0 = 0 and F1 = 1
- 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ...
- Used extensively in computer science
  - pseudorandom number generators
  - sorting algorithms
  - compression algorithms
  - data structures (e.g. Fibonacci heap)
  - ...

Fibonacci Implemented Recursively

```
#include< stdio.h >

int fibonacci(int);

int main()
{
   int i=4;
   printf("%d\n", fibonacci(i));
   return 0;
}

int fibonacci(int n)
{
   int i,j,sum;
   if ( n == 0 )
      return 0;
   else if ( n == 1 )
      return 1;
   else {
      i = Fibonacci(n-1);
      j = Fibonacci(n-2);
      sum = i+j;
      return sum; }
}
```

Var Name	Before	After
i	0	2
j	0	1
sum	0	2

Figure 2.8: Source code plugin in a MindXpres [3] presentation

Some other tools today already take advantage of the aforementioned features, such as HyperSlides [1], which links slides to follow a given storyline or Microsoft PowerPoint, which includes a presentation view with the time since the beginning of the presentation.

NextSlidePlease [2] touches on the same subject with its two lasting points, **Content reuse and Path suggestion** and **Time management**. Having the ability to generate paths and connections between slides automatically and control the presentation time are adequate contributions for a more efficient and planned authoring technique.

HyperSlides [1] also tries to reduce the authoring time by giving an idea, which we discussed in the navigation section, referred to as *Planning with Points*. This subject resurfaces because it forces the creator to produce a storyline before committing to the actual slides, saving time while authoring.

Besides the authoring time, the creator needs fast and consistent styling of the overall presentation. That is something HyperSlides [1] also converges to with another key point,*Styling as a service*. It infers that it is necessary to create slides generated from scenes (high hierarchical points) defined in the storyline and based on predefined design principles.

Most presenting tools already make available generic styles and templates for presentations, but never related to the content of the actual presentation and might be useful in developing a styling mechanism in the future.

Considering that new slideware applications often require different formats and displaying mechanisms, there is an increasing need for new tools to offer a technique that takes other presentation formats and translates them into the one the tools used. This a deal-breaker for new users considering that having to change previous presentations to new formats manually can be time-consuming.

Some aforementioned applications such as SlideDeck.js [6], or MindXpres [3] use custom formats as a way to generate presentations but do not refer to the automatic translation of previous content made on other platforms. There is an article [8] that makes use of the **Reveal.js** framework, a custom external tool that translates PowerPoint presentations to HTML to get the same presentation available on the respective application. It might be a useful guide, considering the value of such functionality and that no other previous solution refers to this problem specifically.

It is also important to mention another part of document authoring, which is the ability of collaboration between different people on the same project, and particularly for this case, to work on the same presentation. It has been one of the main focus of the Google Docs [9] platform that allows several users to author the same document (document, slides, or spreadsheets) at the same time and possibly to use different technologies (computer, tablet, phone, et cetera). Consequentially, the issue of having different versions and merging them or having to wait for another peer to finish the work seizes to exist. A paper

[8]https://olivier.barais.fr/blog/posts/slides/\protect\unhbox\voidb@x\bgroup\defReveal.jspowerpoint/migration/2016/11/30/From$_PPTX_{T}o_{R}eveal_{T}o_{G}ithub_{P}age.html$

[9]https://www.google.com/docs/about/

about collaboration in the cloud at Google [18] tries to find if the collaboration software matters in a team project and the results suggest the tool helps employees collaborate. However, it is not the only tool that acknowledges the need for collaboration in work environments. Webstrates [19], a tool that allows users to share and edit dynamic media content with other users to implement the same functionality but having dynamic content as the main focus of the documents and thus making the line between document and application blurred.

2.5 Discussion

Although one or two presentation applications still dominate the current state for slideware applications, some aforementioned proposals try to tackle many of the issues they pose. In the following points, we will try to summarize the information and discuss whether the solutions found can address the several highlighted problems:

- **Navigation** - The most discussed issue with navigation in standard presentation applications falls into the linearity of slide-based technology. All the section's applications mention that it is essential to address the lack of planning before committing to authoring slide presentations and, consequently, making available a solution that requires the user to predefine a correlation between slides. Non-linear navigation is then used to connect slides that do not necessarily come one after the other but demonstrate a clear contribution to the so-called storyline. Other tools, such as **Reveal.js** or MindXpres [3] allow for extension in the form of plugins as a way to tackle issues like navigation without making the user dependent on a single style;

- **Interactivity** - It has been the most addressed topic, considering that many of the solutions realize the need for changing the static environment from standard presentations. There are many ways to produce slide-based presentations with interactive content, such as interactive data visualization, code editing/execution, and server-client presentations. They all enhance the presentation experience. MindXpres [3] already presents us with two of those interactive functionalities and provides a plugin architecture like **Reveal.js**, so more creative ways of interacting with the audience can be more easily implemented;

- **Authoring** - Authoring is one the best capabilities of standard presentation software, as they are optimized to the end consumer. However, they can still be improved, as shown by NextSlidePlease [2] and HyperSlides [1], using a storyline with authoring techniques based on linkage between different slides.

We can observe the state of the art of alternative presentation mechanisms by analyzing the aforementioned concepts. There is an increasing need for an application more agile and interactive than the field of presentation software currently presents.

HyperSlides is very thorough when it comes to the navigation and authoring topics. It aims to escape linear navigation and improve the authoring experience by requiring a top-level storyline. However, it lacks on the topic of interactivity, which happens to be the main focus of this project. NextSlidesPlease [2] also focus more on the first two topics by showing other ideas to approach them, such as using mental concepts or using several whiteboards in a presentation, but again not discussing how to interact more with the target audience.

SlideDeck.js [6] shares some key objectives with what we want to develop, such as the conversion of some arbitrary language to HTML or interaction with code editing and display through the use of the HTML tag *iframe*. It does so with only basic functionalities. It does not refer to other key components, such as dynamic navigation (with the standard slide presentation format), full conversion from other presentation tools, or interactive data visualization.

MindXpres [3] appears to be the tool that offers the most to tackle the aforementioned subjects, although not presenting a public release to explore further the possibilities stated in the respective articles. The two aforementioned plugins for this platform [4] [8] are great examples of concepts when interacting with the audience and show that a more interactive focused approach is something desired by the industry.

On the other hand, **Reveal.js** provides an open-source framework with some solutions for the problems related to linear navigation and client-server based presentations. For example, extend the current functionality with the use of plugins, in the same way as the MindXpres [3] application.

3

Solution

Contents

In this section, we offer a suitable solution that considers all the aforementioned issues. For that reason, we will define requirements and use them to design a coherent solution with the information from the state of the industry's art. The solution focuses on lecture-style presentations since the project is being developed in an educational environment, making the subject of interaction the most important of the three characteristics mentioned in the related work.

3.1 Requirements

To design a suitable solution, we have to define the objectives based on the description of the problem. In the following subsections, we are going to categorize them with the respective description while ordering them by relevance:

- **Translatable** - Since developing a user interface from scratch would require a workload that does not match the one from this project and also happens to be outside of its main purpose, utilizing a well-established interface to structure a given presentation seems like the logical step to take. It will make the learning curve of the new application more accessible while providing the possibility of reusing previous content with added functionality without having to change it entirely to a new format. Furthermore, finding a structure that does not diverge from current presentation tools and provides the user with common up-to-date techniques while having interactive functionality is also a request feature for this platform. The end-user must not have to learn a completely new environment to use interactive features. Instead, they must use them as new possibilities in standard presentation formats. Therefore, the user must use a standard presentation application and transform it into a new format without learning entirely how the new format works.
- **Interactable** - Being able to interact with the audience, whether by gathering the answers of a question without escaping the presentation environment, receiving instant feedback, or utilizing external resources inside the presentation view, is crucial for the desired application. It must help the user interact with the target audience without restricting or stopping the presentation's flow, i.e., without forcing the presenter to exit the presentation view. It must also guarantee that the user can use different interaction forms without being dependent on external factors. That is, the presentation application must display the content if it does not require an external connection to the internet.

3.2 Foundation

To satisfy all the aforementioned requirements, focusing heavily on the interactivity mechanisms, we decided to work with a platform that fulfills the necessary basic functionalities in a presentation environment. It provides a slide-like structure with customizable styling, highly configurable, accessible (open-source and well documented), and, most importantly, extensible by using a plugin structure. We think this is the most suitable strategy since creating a presentation application from scratch would considerately diminish the ability to produce the interactivity extensions in time, which is the project's main focus. For that reason, we chose to take advantage of the presentation platform **Reveal.js**.

This JavaScript framework offers basic functionalities and grants other features that help develop interactive environments. The most important feature is creating plugins, which we already stated in the related work as a good technique for extending functionality. It will allow us to implement all the interactive features as plugins, making extra functionality more easily implemented, and extensible should new ones emerge in the future. The following statements are major factors for choosing the aforementioned framework:

- It is developed for the web (HTML, Javascript, and CSS), which allows for easy usage on any compatible browser, and integration of new functionality also becomes accessible;

- It has grown in popularity over recent years, gaining a large community while becoming the standard in web-based presentations. It is crucial to ensure the impact and longevity of the proposed solution;

- It offers the possibility for multiplexing, allowing the audience to view the presentation slides the creator is controlling on their own mobile devices such as a portable computer, phones, tablets, or any device with an internet connection and access to a web browser. It employs a Socket.io server (real-time application framework for bidirectional and event-based communication) and is going to be useful when implementing features of audience interaction;

- Extra functionality is possible due to a plugin architecture, which highly enhances the extensibility possibilities;

- It is registered under the MIT license, allowing for more flexibility when adding new functionality.

While the points above state the strong features of the proposed framework, they do not specify the rationale for excluding the aforementioned applications, such as Microsoft PowerPoint or Google Slides, that also provide a plugin or add-on environment. Both tools require the user to connect to their respective services, i.e., having an account. Although having these accounts is somewhat trivial nowadays, the idea is to grant the creator as few restrictions as possible and establishing the possibility of adding more functionality more readily.

All the above statements establish the proposed framework as the best candidate for the solution's core.

3.3 Architecture

The new features are joined with the core application by designing a specific plugin for each feature. A **Reveal.js** presentation works like a typical front-end application, where the user has the main HTML file, the corresponding Javascript and CSS files. To manage other useful features more efficiently, such as connections between servers and clients, SCSS compilers, and task runners, we will use Node.js [1] in these specific situations.

. Furthermore, **Reveal.js** offers the possibility of adding plugins to increase its functionality. To accomplish that, it registers the plugins before initializing a presentation. They are be grouped in a bundle alongside the translation module (DynaSlides project), which is responsible for registering all the plugins and handle all the necessary interactions between the foundation, the translation module, and the respective plugins. As shown in figure 3.1, the Dynaslides bundle works by providing a Microsoft's Powerpoint file to a translation module using an user interface. The translation module will then translate the file to the new format and while doing that, it will also search for the plugins and consequentially create them in the new format.

[1] https://nodejs.org/en/about/

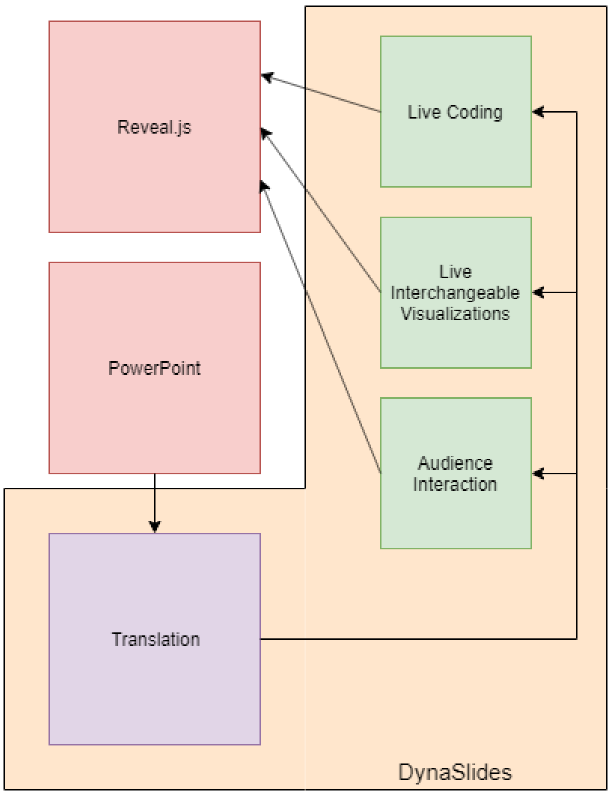


Figure 3.1: Architecture structure for the DynaSlides bundle

3.3.1 Final Presentation Directory

. Since the Dynaslides presentation format integrates several web technologies and frameworks, such as HTML, CSS, Javascript, and **Reveal.js**, the final document is a compressed .zip file that contains the necessary files and directory for the presentation to work. Namely, as shown in figure 3.2:

- **data** - This directory is used to store the files gathered from the live visualization plugin we will discuss later on.
- **images** - This directory holds image files used on the presentation. The application might embed some of the images in the index.html file as well.
- **js** - This directory holds javascript documents used for the presentation. Some javascript is also embedded in the index.html document.
- **reveal** - This directory holds all the necessary file to execute a **Reveal.js** presentation. It is taken from their official git repository, and this is where we can add more plugins made for **Reveal.js**.
- **index.html** - This file holds all the HTML, Javascript, and CSS code besides the ones already present in the aforementioned directories. It also contains embedded images and other attributes converted from Microsoft's Powerpoint.

We first wanted to provide one simple HTML file for all the presentations, but that became rather difficult since some presentations may have different content present in them. This means that those presentations would take a considerable amount of time to open and could even impact the performance during the presentation.

For that reason, we decided to only offer the presentation in the aforementioned format.

There is one exception to this when the presenter wants to have audience feedback. We will further discuss this topic in the plugin section 3.6.

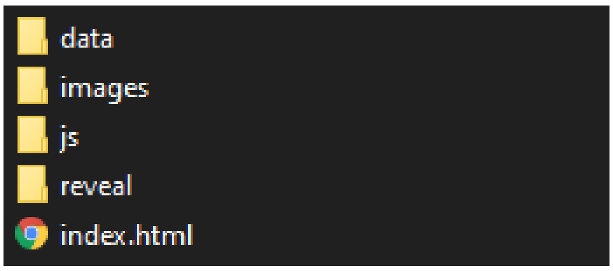

data
images
js
reveal
index.html

Figure 3.2: Dynaslides final presentation directory

3.4 Implementation

To start developing the desired application, taking into account the requirements mentioned above and the foundation that we chose to use for our final presentation environment, we first needed to solve three glaring problems.

The first problem was to find a suitable solution to translate a Microsoft's Powerpoint presentation to our desired format (HTML/**Reveal.js**). We decided to start with a basic software we found on the related work section 2, which already had the framework to translate Microsoft's Powerpoint documents into a **Reveal.js** presentation. Although it was incomplete, it allowed us to save some time on this complex solution that is translating the presentation and use that time to work on the plugins, which are the main focus of this project. We will further discuss this starting software in the translation section 3.5.

The second problem was to find out how to implement these plugins into the new foundation, which is **Reveal.js**. We decided to take advantage of **Reveal.js**' plugin structure and use Node.js to easily implement the plugins. The Dynaslides bundle offers some basic plugin functionality, and each plugin adds to that same functionality. We will further discuss this starting software in the plugins section 3.6.

Finally, we had to decide how to offer this new application to the user. We decided that it would be important to offer two different approaches based on the user's software skill. We will be further explaining the different approaches as we discuss the features implemented in the Dynaslides application.

3.4.1 Website version

This version is intended for the user with no experience with a command-line interface and wants to translate the presentation more easily. It is built like a normal website, i.e., it has a back-end/front-end configuration and also a server that can be used for the audience interaction plugin 3.6.4.

The back-end takes care of all the features mentioned in the architecture, such as the translation and the plugins present in the translation, and then sends it to the front-end to be downloaded as a .zip file. This is done using the express module provided by node.js, which quickly allows creating an endpoint so that the front-end implementation can communicate with the back-end implementation.

The front-end gives the client an easier way to translate the presentation, with the different required options. It will inform the user about the translation state and provide an interface to control some features used by the plugins available. This is implemented using the React Javascript framework, which allows us to create the user interface and the respective communication with the back-end more easily.

As we can see in figure 3.4, the tool offers simple HTML buttons that can translate the presentation with the help of the reference guide and the custom settings necessary. The presenter can press the additional settings text available in the aforementioned figure to open the settings box and see the figure.

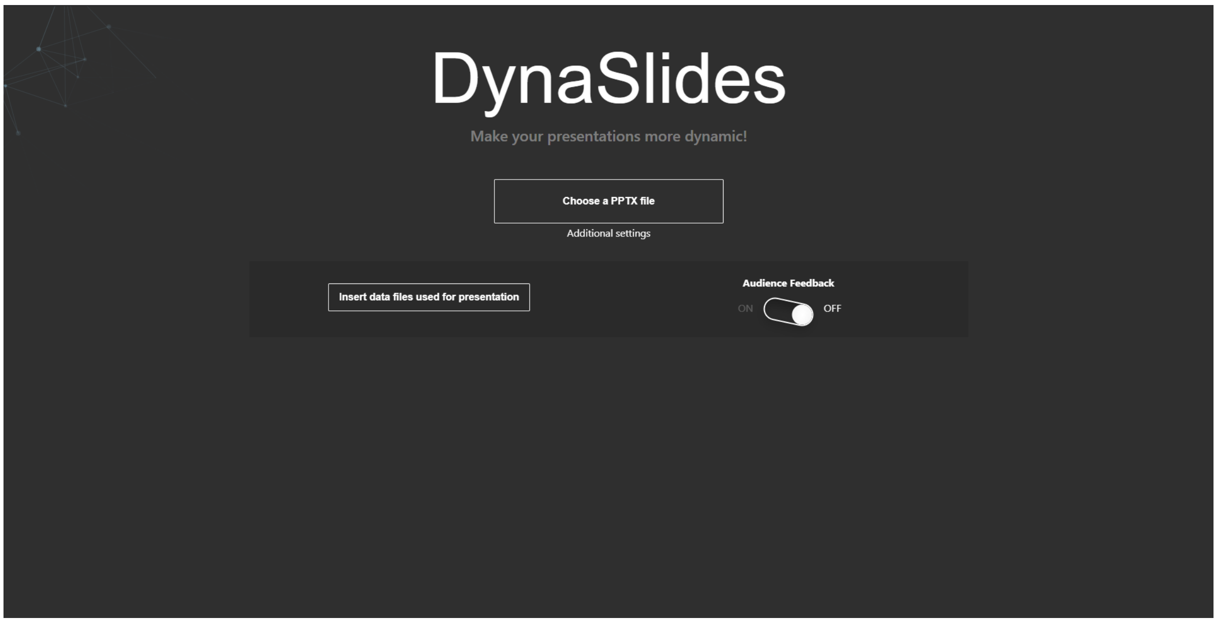

DynaSlides
Make your presentations more dynamic!
Choose a PPTX file
Additional settings
Insert data files used for presentation
Audience Feedback
ON
OFF

Figure 3.3: Dynaslides website environment before uploading the presentation

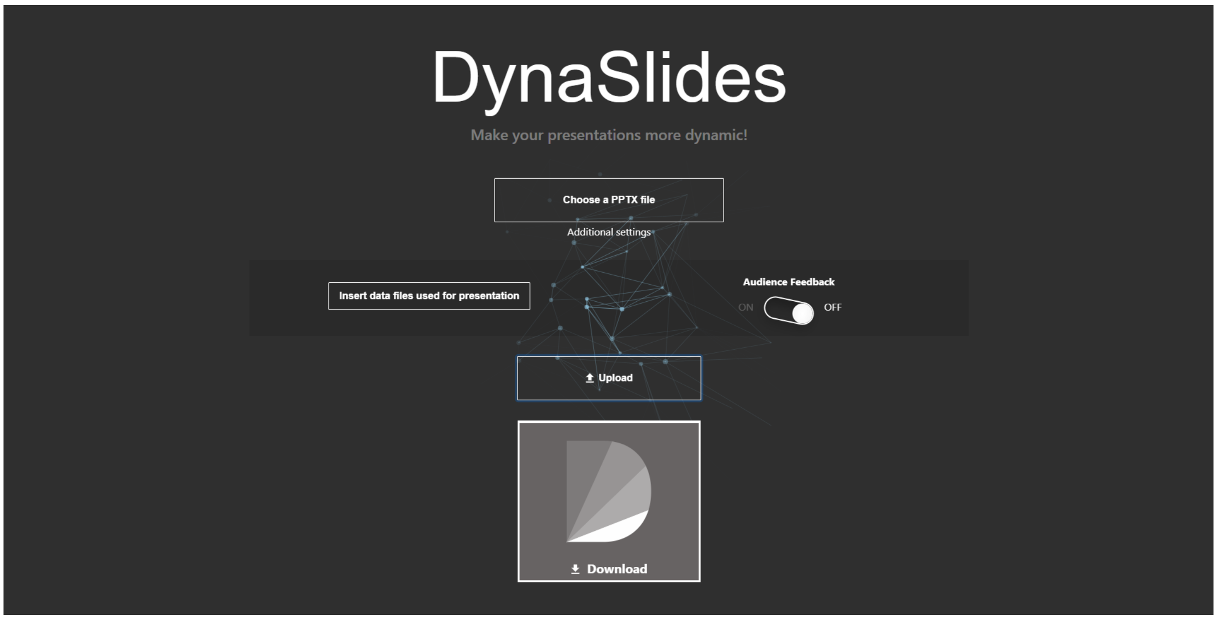


Figure 3.4: Dynaslides website environment after uploading the presentation

Finally, it also offers server code that needs to be set in a publicly accessible environment, such as a public endpoint, to provide the live feedback that the audience interaction plugin requires. The clients will be able to see the presentation on their respective devices using the public endpoint. They will also be able to interact with the presentation shown by the presenter. The presenter must be the one to set up this endpoint, and the execution is explained in the audience interaction section3.6.4 and also in the reference guide of the final application.

3.4.2 Command line version

This version is for users acquainted with a command-line interface and prefer to handle the different available options. It has the same functionalities as the website version and is also explained in the application's reference guide.

The command-line interface uses node.js to run the application that is developed in javascript. The input is as follows:

```
node dynaslides.js <pptx> <csv1 cvs2 ...> <options>
```

The application will try to find the Microsoft's Powerpoint file, then the data files associated with the live visualization plugin, and finally the options. For that reason, if no Microsoft's Powerpoint file is made available or a data file is missing, the application will launch an error and stop the execution. We will further explain the functionality in the translation section.

Finally, both methods can be used using the code provided in the git repository of the project. This execution is well explained in the respective reference guide.

3.5 Translation

As we stated before, the translation module started with an already working solution[2] that translates Microsoft's Powerpoint presentation to a new HTML format, which then uses the **Reveal.js** presentation framework.

Although the starting code allowed us to work more on the plugins we made available, it was far from a complete translating solution. The application only translated some characteristics from the original presentation, and some elements were not correctly positioned. It lacked the ability to translate themes with their respective backgrounds, it could not find elements that were related with the other components of the Microsoft's Powerpoint hierarchy, i.e., the slide master and the slide layout configurations, and many attributes of certain elements were not implemented, such as image effects and font elements. Moreover, the tool was only available to use in a browser environment, which diminished our options and flexibility and even the application's performance since it could not take advantage of the resources the client's computer had to offer.

Therefore, we first started by changing the application so it would work with node.js. The previous version would take advantage of *web workers* which are a simple implementation for web content to run

[2] https://github.com/g21589/PPTX2HTML

scripts in background threads. Since these workers are not required and are not the best solution in node.js, we used the node.js package system to implement the same functionality it had while working on the browser. This change also enabled us to have a command-line version and a more organized version of the application. It now uses the node package manager or npm for short.

After that, we started by reviewing what was already done and the current problems. To do this, we took advantage of the *Office Open XML* presentation reference guide [3] which gives us information on how to read and use the free information available in a Microsoft's Powerpoint document. In this case, the file can be read as a .zip file made of particular components or, more specifically, directories with .xml files.

The structure is organized according to the *Open Packaging Conventions*, and the Dynaslides application focus on the two most important directories:

- **_rels** - This directory holds the relations between the different available documents in the current directory. It allows us to produce a table to address those relations rapidly. There is one per each directory in the .zip file. This implementation was already working as intended in the starting solution and did not need any further improvements.

- **ppt** - This directory holds all the data regarding the presentation. It has six sub directories, namely the `handoutMasters`, the `media`, the `notesMasters` ,the `notesSlides` ,the `slideLayouts` ,the `slideMasters`, the `slides` and the `themes`. Each directory handles a feature of the presentation and retains all the important data to replicate it. Although the starting solution had the data gathering of these directories almost done, the retained data was not being utilized correctly in most cases, such as in font properties, presentation hierarchy and background themes. So our next step was to work on this missing data usage.

Firstly, we began by fixing the hierarchy of the information in each slide. A slide is made from the combination of three components forming the said hierarchy. At the bottom, we have the master slide, which the slide is based upon. It encapsulates default properties of the slide, such as the positions, font styles, background, title, body, and footer. In the middle, we have the slide layout applied to the slide in question, which contains information that overrides the one present in the master slide. Finally, we have the slide itself that contains information not already specified in the master slide or the layout. This information is only used by one particular slide.

Using this structure, we were able to implement each slide's features according to this hierarchy. If a property regarding an element of the slides was not present in the slide, then we searched the corresponding slide layout using the relations table and tried to find the property in that layout. If we still did not find the property in the respective layout, we used the master slide to get the element's default configuration.

[3]http://officeopenxml.com/anatomyofOOXML-pptx.php

After that, we focused on implementing each element's features in HTML. Since there are many elements provided by the original presentation, we worked on the most important features and we tried to make them as close as possible in the new format.

3.5.1 Position

The elements' position is arranged by taking the slide size and the most common computer monitor DPI (dots per inch) to translate the position given in English Metric Units or EMUs to the respective pixel size. The starting application had this system implemented correctly. However, it had to be improved by ensuring the sizes, rotation, margins, and padding of elements were the same as those displayed by the original presentation. To do this, the hierarchy of the elements had to be used to get the correct properties. Then, the resulting properties had to be translated to the respective CSS elements in the new format. To find the elements in the layout or the master, all it took was to use the relations table mentioned above with the element's id.

As shown in figure 3.5, the Microsoft's Powerpoint slide elements are almost identical on the Dynaslides format. The starting translation application could not get the elements' position correctly, inside the shape they were in, such as the case of the `00` text element, because it was not taking into account the centering factor of that element. Moreover, the original slide can not show the text's parent elements correctly since the application does not consider the hierarchy stated earlier.

3.5.2 Shapes

Shapes are the most important element in this translation since almost every object in a Microsoft's Powerpoint is derived from a shape. The starting application already had core shapes implemented, such as rectangles, round rectangles, ellipses, connectors, and basic arrows. However, it lacked the ability to translate custom shapes into SVG elements, such as background themes. It also lacked the ability to recreate some properties of the shapes, such as borders and colors. To do this, the application converts the custom shape elements present in the .xml file of the slide or the respective layout/master components to inline SVG elements in the new format.

With this translation, it was now possible to translate more shapes and even some theme backgrounds that use a custom shape to produce more complex elements.

As shown in figure 3.6, the arrow element, some colored circles, and the title bar do not have their colors correctly assessed. This error is due to the wrong hierarchy analysis made by the starting solution. After we fixed that problem, the shapes' properties were now being shown in the original presentation.

We can see another good example regarding the custom shapes in figure 3.9. In that particular case, the shapes of the background theme's images are well-defined custom shapes. Since the starting

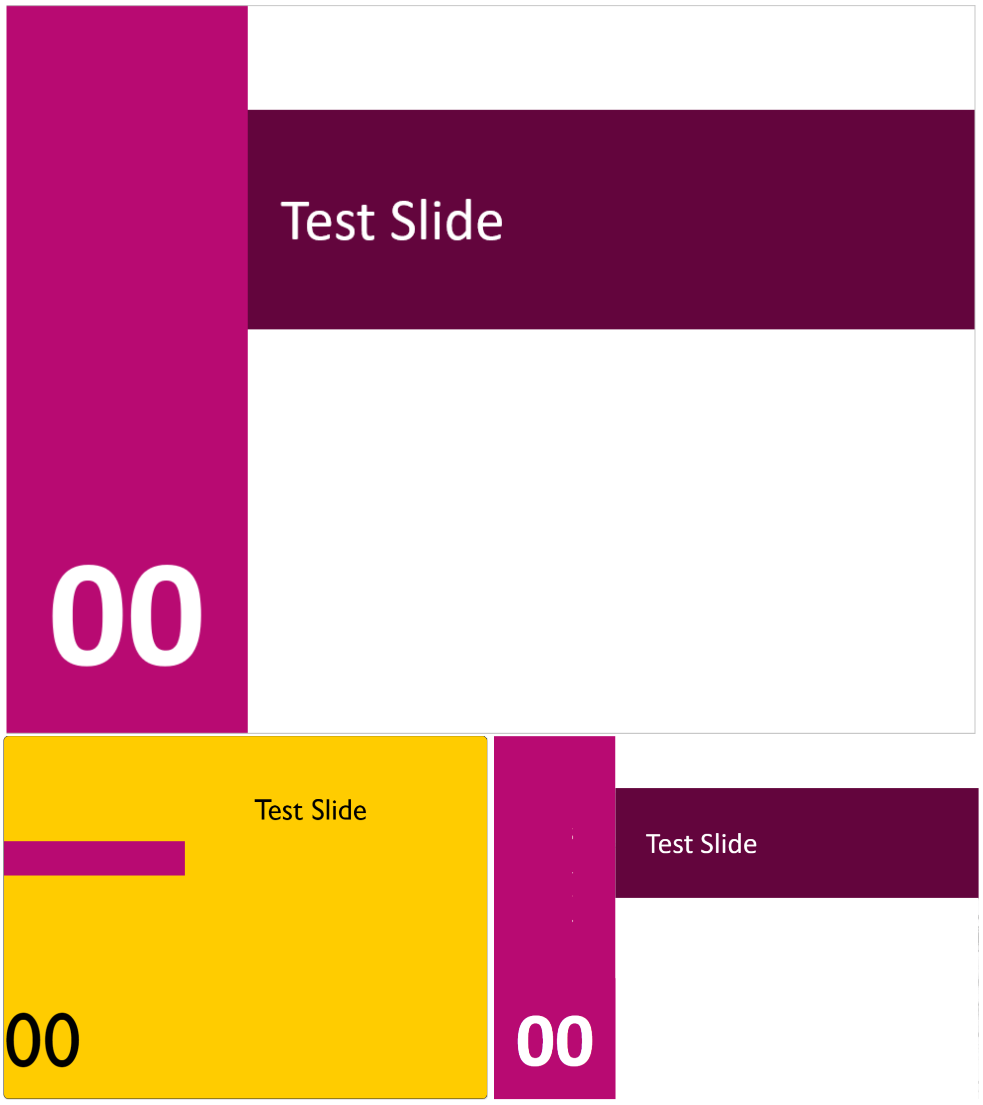


Figure 3.5: Difference in positions from Microsoft's Powerpoint (top slide), starting translation application (bottom left slide) and Dynaslides presentation (bottom right slide).

solution does not include those shapes, they are not shown in the resulting translation. After we fixed the problem, the presenter can now see the images as in the original presentation.

3.5.3 Color

Colors were a big problem to translate since the starting application did not consider the same hierarchy stated earlier. Another problem was that colors could have effects embedded in them, such as saturation, luminosity, alpha, and more. We implemented the effects as referenced in the *Office Open XML* presentation reference guide [4] by reproducing them in the CSS of the new format.

[4] http://officeopenxml.com/anatomyofOOXML-pptx.php

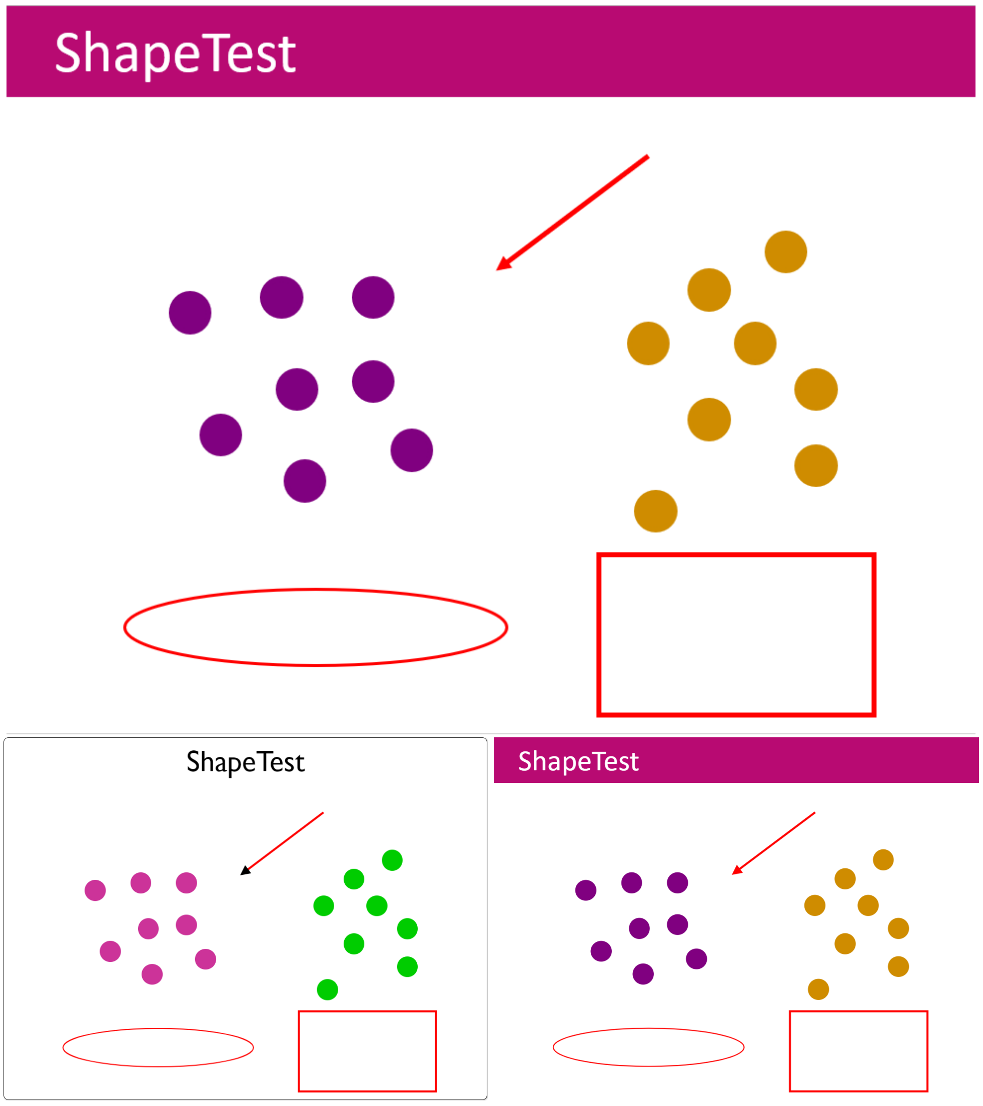
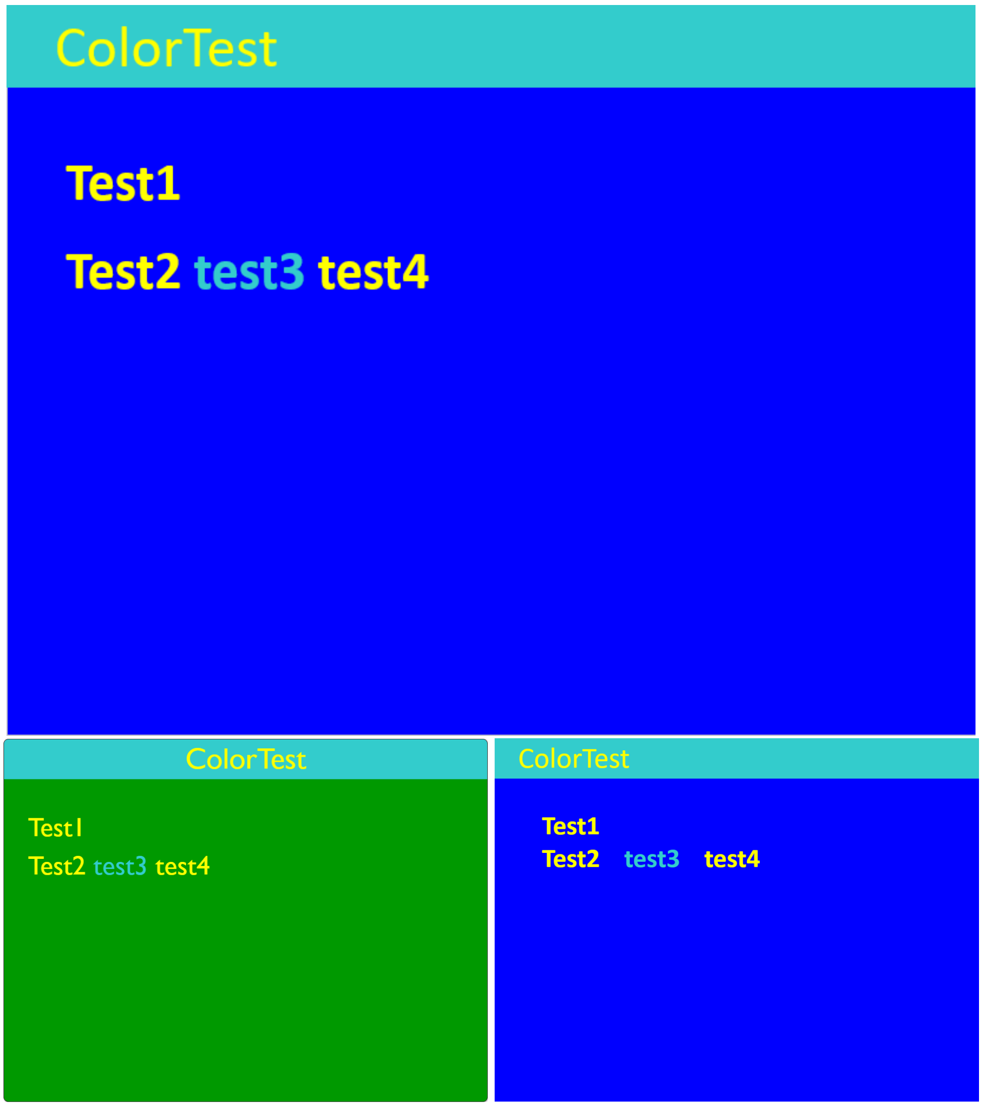


Figure 3.6: Difference in shape properties from Microsoft's Powerpoint (top slide), starting translation application (bottom left slide) and Dynaslides presentation (bottom right slide).

As shown in figure 3.7, the background color of the translation made by the starting application is wrongly assessed due to the hierarchy of the slide shown. The color was not being referenced in the slide file since it was in its respective layout. For that reason, the color displayed is one of the default colors of the theme chosen for the presentation. We can see this wrong assessment throughout all of the other feature examples.

3.5.4 Images

The main problem with images was that the only translated element in the starting solution was the image itself. It did not consider the effects that images can have in the original presentation. We imple-


ColorTest

Test1

Test2 test3 test4

ColorTest

TestI

Test2 test3 test4

ColorTest

Test1

Test2 test3 test4

Figure 3.7: Difference in color assessment from Microsoft's Powerpoint (top slide), starting translation application (bottom left slide) and Dynaslides presentation (bottom right slide).

mented the basic effects, such as duotone, tint, luminosity, and cropping, according to their description in the *Office Open XML* presentation reference guide [5].

As shown in figure 3.8, the Instituto Superior Técnico logo is not correctly cropped in the starting application slide, but instead just resized to the cropped image size. Using a library for image manipulation in Javascript, we could now see the original presentation's cropping effect.

[5]http://officeopenxml.com/anatomyofOOXML-pptx.php

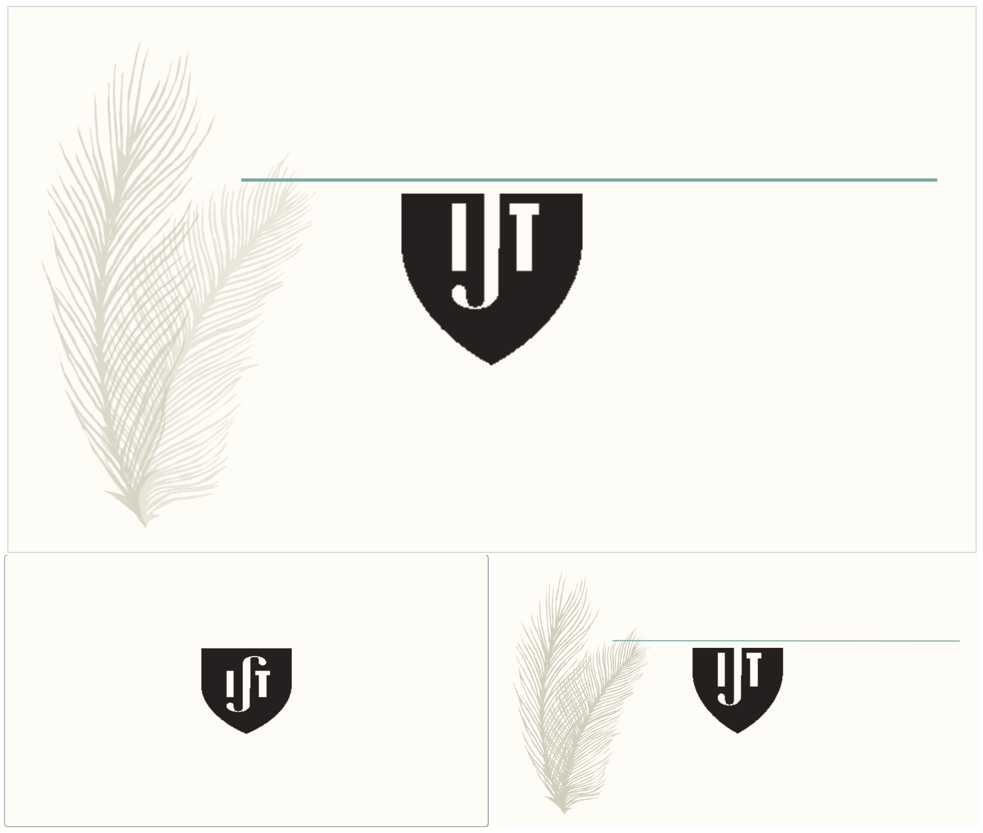


Figure 3.8: Difference in the cropping image effect from Microsoft's Powerpoint (top slide), starting translation application (bottom left slide) and Dynaslides presentation (bottom right slide).

3.5.5 Themes

Themes had a problem related to their respective backgrounds since the starting application did not consider the custom backgrounds that Microsoft's Powerpoint offers. We implemented a way to translate background themes configured as custom shapes that can be converted to inline SVG elements. Moreover, although the data was being gathered successfully from the color scheme, the application did not display some of the colors correctly due to the faulty hierarchy system. When we resolved the hierarchy problem, the colors started being correctly displayed as well.

As shown in figure 3.9, all the leaves are translated using a custom set of points to a new svg inline image. The original translating application could not use this method and therefore it wasn't able to show the leaves in the new format.

3.5.6 Other attributes

Finally, the remaining features that are not discussed in this section (animations, videos and other complex features) will not be translated due to the complexity of translating a Microsoft's Powerpoint

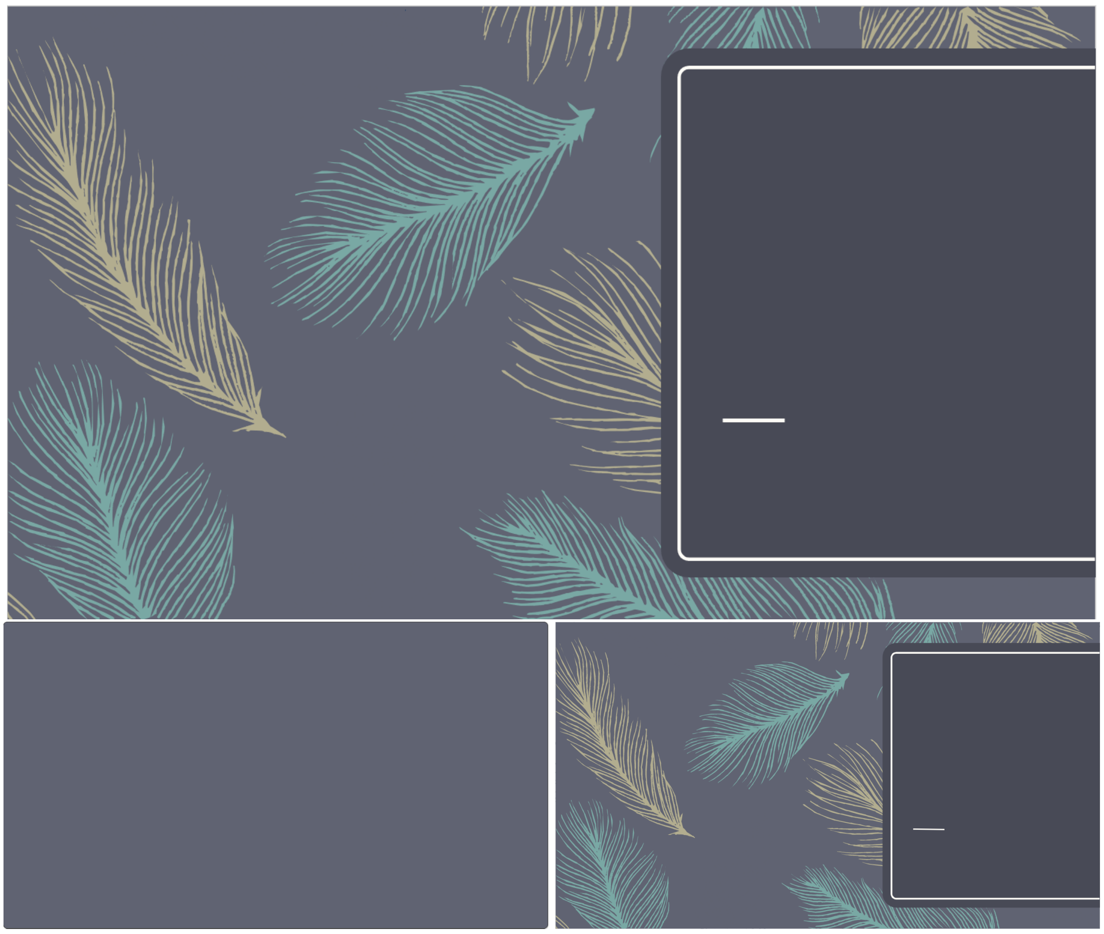


Figure 3.9: Differences in theme background from Microsoft's Powerpoint (top slide), starting translation application (bottom left) and Dynaslides presentation (bottom right).

document. We further explained the reasons for this in the limitations section 5.2.

3.6 Plugins

Each plugin represents the logic behind each interactive feature. They will behave independently and according to the presenter's specifications.

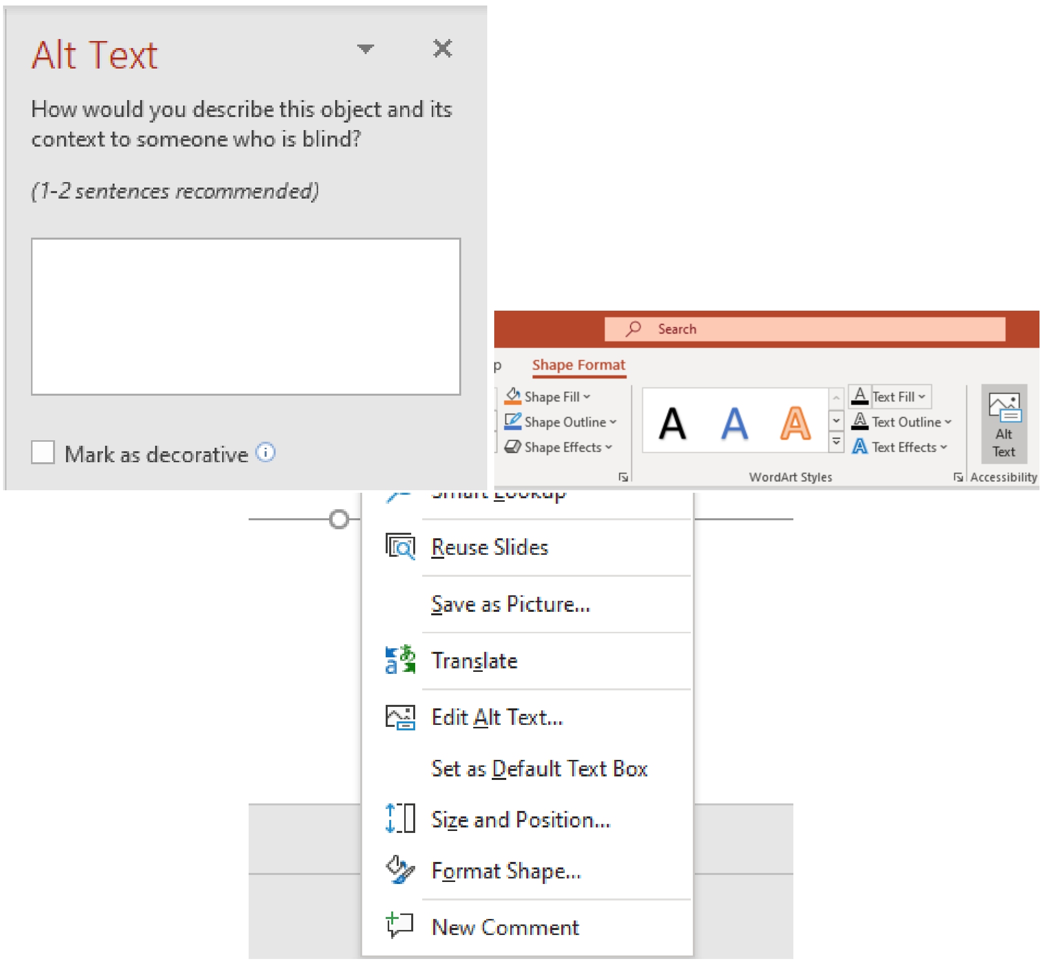
The plugin structure offers basic functionality, common to all plugins, and then that functionality is complemented by the plugins. Each plugin works by using a translated text box from Microsoft's Powerpoint while taking advantage of a property common to all objects in the original presentation called `alt text`.

We first tried to use the text from inside the text box, but this would not be ideal since the text in Microsoft's Powerpoint would be affected, and the original presentation would have the command showing on the text box. We then tried to use the notes from each slide, although this wouldn't work either since there was no way to uniquely identify the text box for a given plugin if there were more than one in the same slide.

The `alt text` property, seen in figure 3.10, is therefore perfect for choosing which text box belongs to a given plugin since it is always present at the slide level of the translation and it is not visible on the actual slide during the presentation. This means we can use that field to identify the plugin and gives us the ability to control the plugins' inputs.

The only downside to this solution is that the description of the alt text is affected. This might be a problem when a long description is needed, but overall, the alt text field is perfect for what we want.

Therefore, we created a function that reads every alt text property available in each text box of a given slide. If it finds a custom set of characters predefined by us (in this case, %%), it will know that the text box is used for a particular plugin. This custom set of characters has custom commands inside, and



Alt Text

How would you describe this object and its context to someone who is blind?

(1-2 sentences recommended)

Mark as decorative

Figure 3.10: Alt text box where the user inserts the custom command (top left figure), selecting the alt text property from the shape format menu (top right figure) or by right-clicking in the text box (bottom figure).

they are explained in each plugin, although we represent them as so:

```
%%<custom command>(<options>)%%
```

This allows the presenter to create a simple text box, and it's content to recreate the interactive plugins as we see fit. We can even use the text from inside the text box and the custom command to create even more functionality, such as in the live coding plugin 3.6.2.

The decision of choosing the %% symbols to encapsulate the commands came down to the fact that the percentage symbol is one of the least used symbols in common text.

3.6.1 Shared Plugin Functionality

As we stated before, the plugins share core functionalities that they will then extend. After finding a text box on the slide, the application will check if the **alt text** property has one of the registered plugin command tags written inside. If it has, the Dynaslides application will inform the respective plugin, and it will create an interactive feature in the place of the original text box.

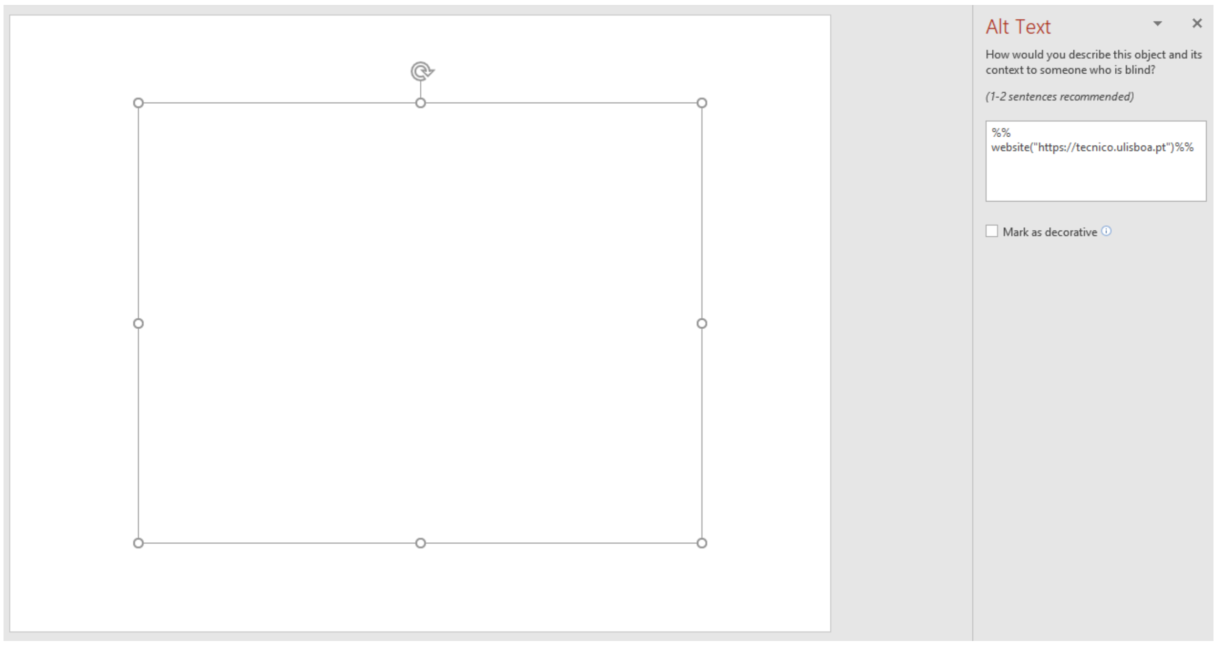
Therefore, the first feature, which is common to all the plugins, uses the custom command system represented above to allow the presenter to show a website of his choice while on the presentation. As shown in figure 3.11 on the top, we can see the custom tag used is `website`. It also has an option, which is the link to the actual website. It will use the text box's dimensions in the original presentation to create an HTML `<iframe>` that shows the website on the slide itself, such as in the previous figure on the bottom. This feature will consequentially require an internet connection to display the respective website.

Another feature shared among all plugins is the ability to have audience feedback. This feature will impact how the file presentation document is constructed.

To first activate the feature, the presenter must inform the Dynaslides application that the final presentation must have audience feedback. This can be done by either activating the audience feedback button seen in the additional settings of the website version, as seen in figure 3.3 or by providing the option `ai_on` in the command-line version.

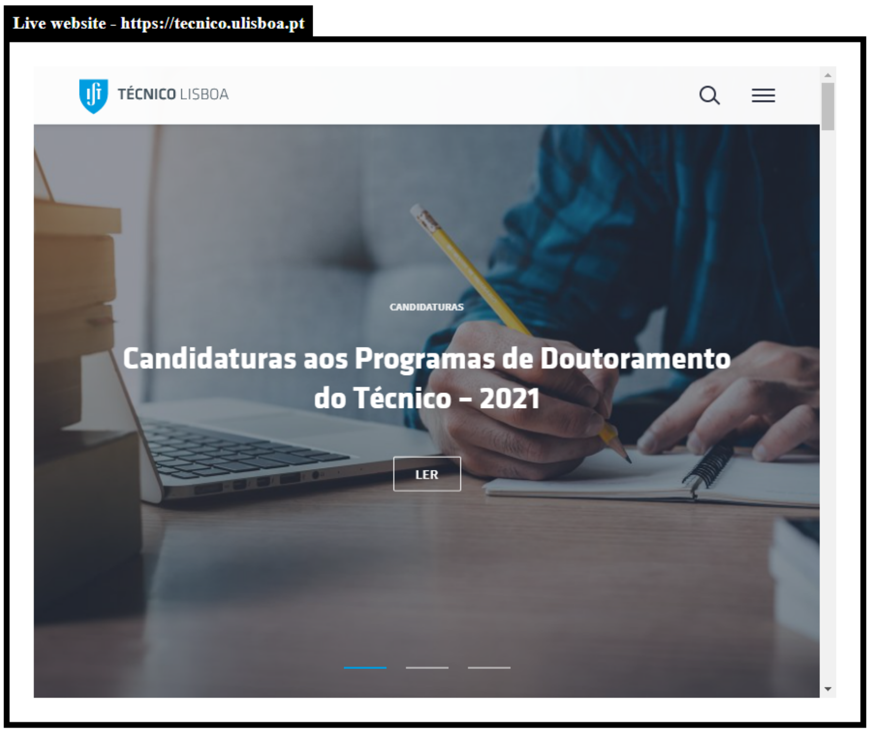
After activating the aforementioned option, the application will create two directories instead of one in the presentation's final .zip file. One directory will correspond to the audience's view and enables them to interact with their own devices, i.e., the client version. The other directory will correspond to the presenter's presentation, shown on a big screen for all of the audience, i.e., the master version.

The client version is provided with an interface so that the user can interact with the presentation on the device of choice. Besides giving the ability to change through all the slides available, the interface gives the ability to rate or evaluate the presenter's slides. As shown in figure 3.12, a presentation's attendee can give a like or a dislike to a given slide and even send a custom message to the chosen



Alt Text
How would you describe this object and its context to someone who is blind?
(1-2 sentences recommended)
Mark as decorative

%% website("https://tecnico.ulisboa.pt")%%



Live website - https://tecnico.ulisboa.pt
TÉCNICO LISBOA
CANDIDATURAS
Candidaturas aos Programas de Doutoramento do Técnico – 2021
LER

Figure 3.11: Website plugin features in Microsoft's Powerpoint (top figure) and the respective translation in the new Dynaslides format (bottom figure).

slide. The attendee can only give one like/dislike per slide but can send one or more messages per slide.

The master version shows the presentation to the public, and when the presenter changes from one

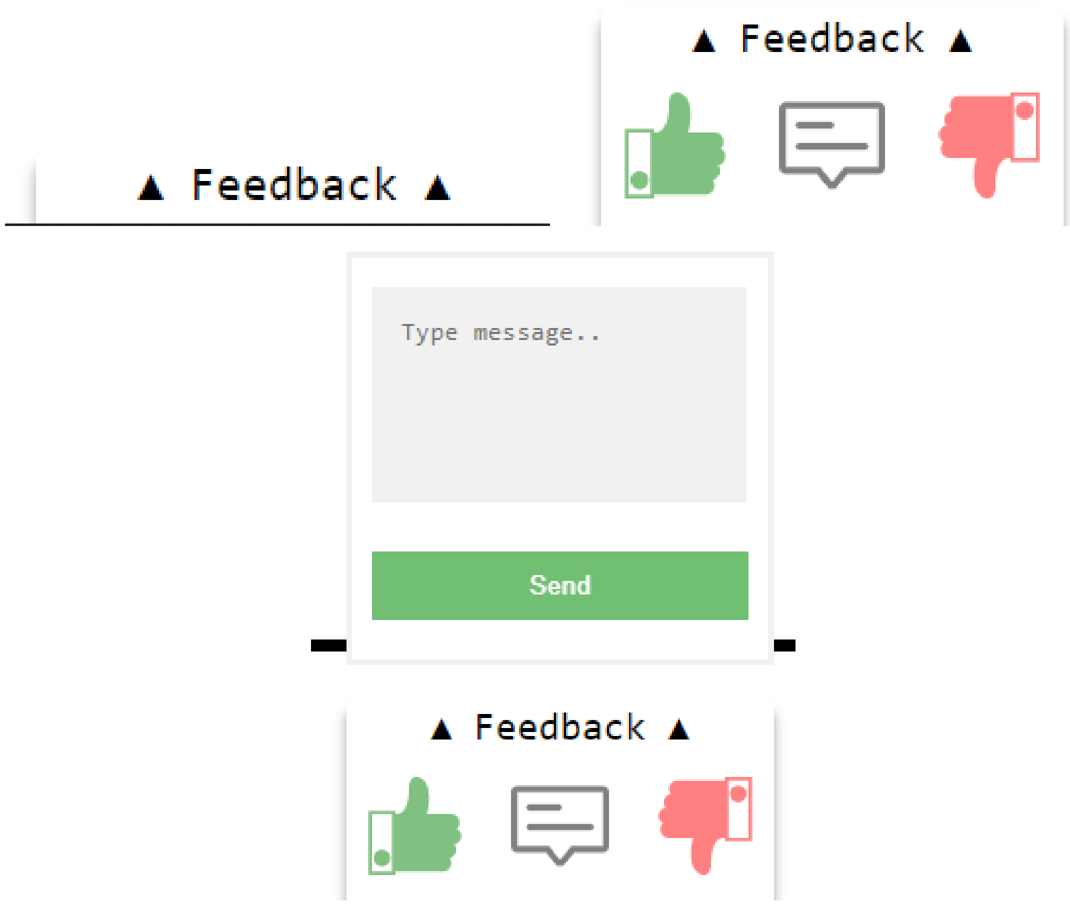


Figure 3.12: Live audience feedback menu in the client's view before opening (top left figure), after opening (top right figure) and with the message box opened (bottom figure).

slide to another, all the client presentations will change to that respective slide. When the presenter ends the presentation and closes the browser tab where the slides were being displayed, the browser will download a .json file with all the feedback information retained from the public.

The aforementioned functionality is possible due to the multiplex plugin provided by **Reveal.js**. This plugin provides the logic of the two different presentation formats (client and master), particularly the client version's logic showing what slide the master version shows. The Dynaslides application extends the process by proving the aforementioned functionality of audience feedback and the consequential interface. All the implementation is possible due to the **Socket.io** Javascript library, which enables real-time, bidirectional, and event-based communication.

However, the aforementioned process requires two additional settings to be used.

The first one is that there must be an internet connection during the presentation since both versions need a way to communicate. The presenter has to place the client version and a server file in a publicly accessible endpoint to give the attendees access to the client presentation on their devices. The presenter can use the master presentation locally, but it needs access to the previously mentioned endpoint to communicate with the client presentation. All this process is detailed in the reference guide of the final prototype.

The second setting refers to an authentication system that prevents duplicate information coming from each client. To do this, the Dynaslides application creates a cookie in the browser, of each client, with an id and a fixed time span (provided by the presenter). Each request for the master presentation will then use this id. For example, if the client tries to rate a slide already rated, the master presentation will ignore that information. Although this solution is by no means the best authentication system, since the client can delete the cookies or use a different browser, a more complex authentication system would require the user to register or the application to gather the target user's IP. We explored those options, but eventually, they were either too complex or would ruin the presentation's experience overall.

The solution made with the following plugins is based on the aforementioned shared functionality. Moreover, all of them have access to the text box's elements in the original presentation.

In the next sections, we will further explain each particular plugin structure, how they execute the particular requirements, and how the presenter can use them.

3.6.2 Live Coding

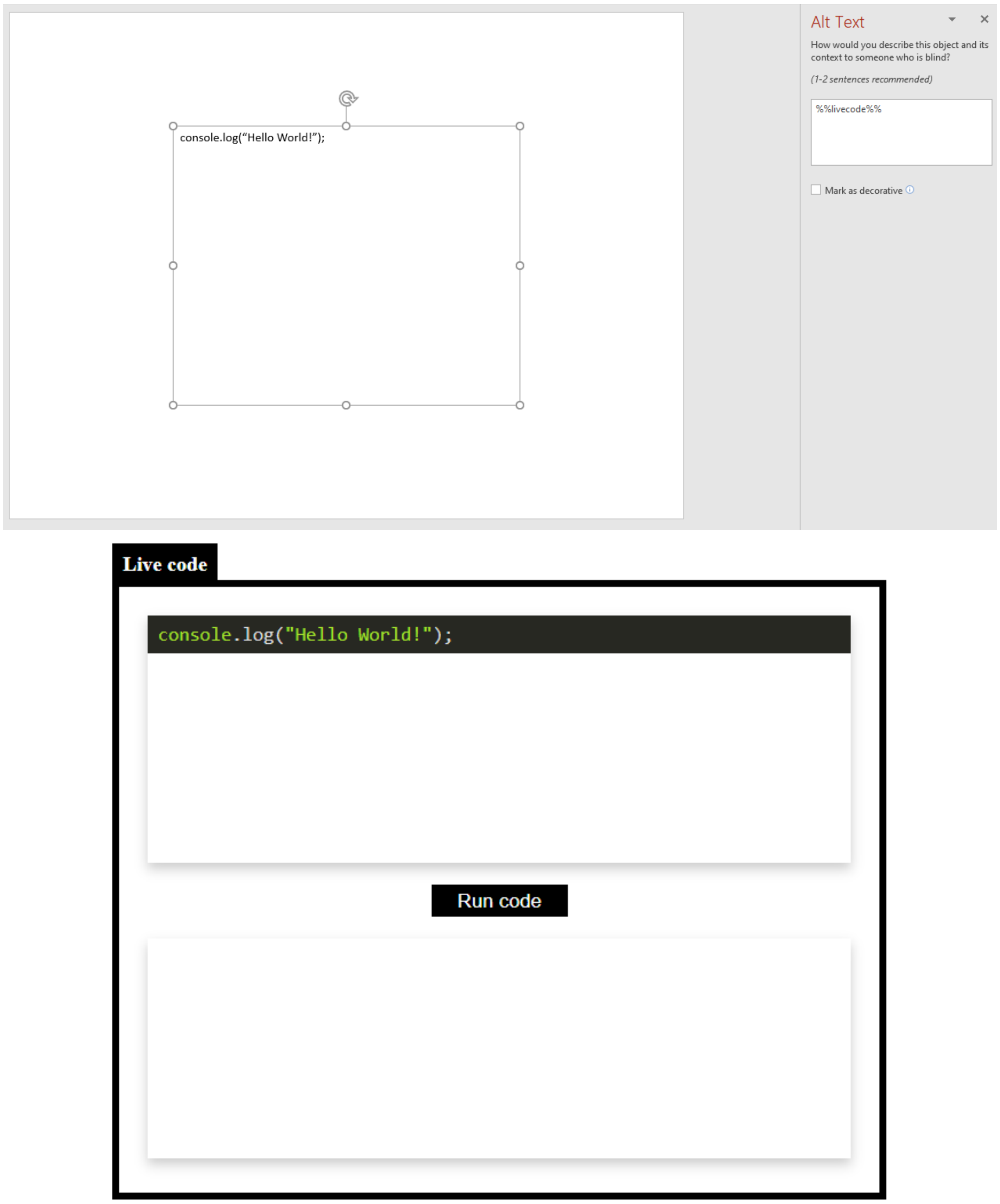
The live coding plugin consists of displaying, executing, and editing code inside the presentation view.

The plugin works by gathering the code as text inside the text box and using the same system shared by all the plugins. As shown in figure 3.13, the presenter sets the code, which in the current version of this plugin is only available as Javascript, and then sets the command tag `livecode` in the alt text box. The reason why we chose to use the text inside the text box, in this case, is because the presenter probably wants the keep the text formatting the same for the code. Therefore, since the space is limited in the alt text feature, we decided to use the text from the actual text box. This also presents another advantage for the user. If an error occurs and the application can not translate the code, the presenter will always have the code available to show in the original presentation.

The translated code is then placed in a `div`, in conjunction with a button to execute the respective code and with another `div` that will show the results of the execution.

The HTML `div` that contains the code uses a **Reveal.js plugin** to highlight the code, which will make the code much easier to perceive. This container also provides the ability of altering the coding during the presentation.

The execution button uses a Javascript function called `eval`, which takes the code and runs it, like the console present in most browsers. Since this function can be dangerous if the user does not know or



console.log(“Hello World!”);

Alt Text

How would you describe this object and its context to someone who is blind?

(1-2 sentences recommended)

%%livecode%%

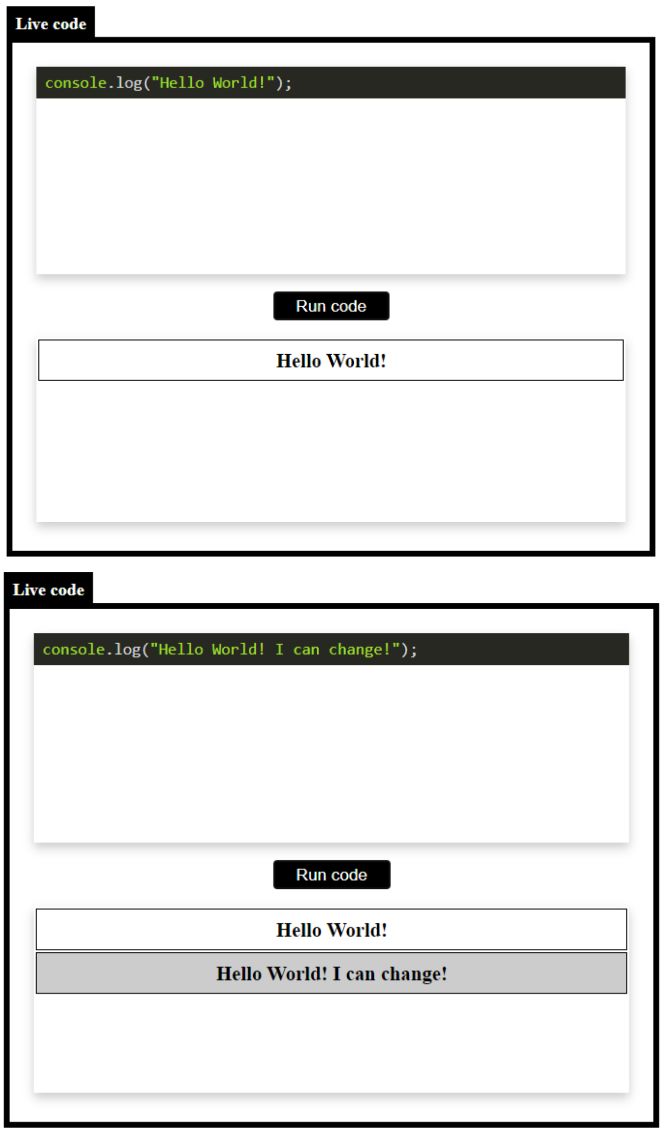
Mark as decorative

```
console.log("Hello World!");
```

Figure 3.13: Live coding plugin features in Microsoft’s Powerpoint (top figure) and the respective translation in the new Dynaslides format (bottom figure).

understand the output of the code, a message will appear and ask the presenter for confirmation before running the code.

Finally, the last HTML `div` of the plugin will showcase the result from pressing the button. As shown in figure 3.14, on the top image shows the execution of the code displayed in the bottom image of figure 3.13. On the bottom image, it shows the edited code and the consequential execution.



```
console.log("Hello World!");
```

```
Hello World!
```

```
console.log("Hello World! I can change!");
```

```
Hello World!
Hello World! I can change!
```

Figure 3.14: Differences after running the code presented in the live coding plugin (top figure) and after changing the code and running it again (bottom figure).

The live coding plugin happens to be only used by the presenter to showcase scenarios related to code manipulation. Therefore, it does not need to use any of the aforementioned shared functionality.

3.6.3 Live Visualization

The plugin consists of creating an interface where the user can change how the data is visualized. The presenter can add, edit, or remove data while remaining in the presentation environment. The process starts by providing a path to the data file with a predefined format (.cvs or .json). To do this, the plugin uses the same shared method as the other plugins. As shown in figure 3.15, the plugin uses the tag `liveviz` and the options that come with it, which in this case is the name of the data file. It will use this data file and the text box's position and size to generate an interactable visualization during the presentation (the presenter can change how the data is displayed and also add/remove data from the visualization during the presentation).

We chose to use a simple data file to populate the visualization, where the data has two dimensions forming a point, i.e., (x,y). For example:

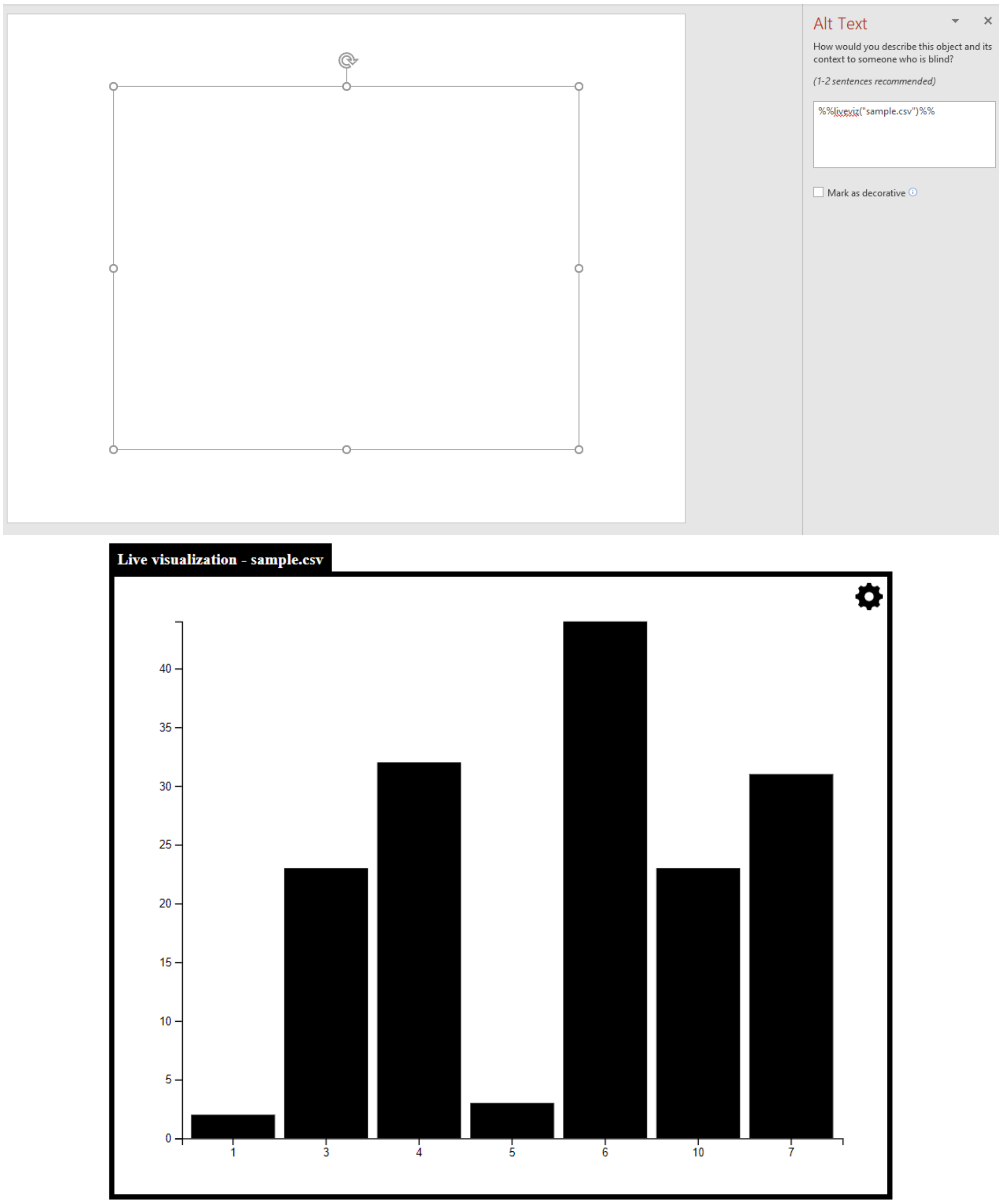
x	y
2	4
3	5
5	5
6	7

Due to the complexity and amount of different datasets available, we decided that we would only use this simple format to show it's possible to interact with visualizations while performing the presentation. Any other datasets that are not two connected dimensions and not in the .cvs or .json format will generate an error.

During the presentation, the presenter will have access to a menu that offers the possibility of choosing between a limited set of different visualizations according to the data provided, much like the one discussed in the related work. The presenter will also have access to another section that allows adding, editing, or removing data during the presentation. As shown by figure 3.16, when hovering the settings gear icon on the top, the other two icons will appear.

The first icon (two arrows crossing each other), when hovered, displays the three red squares next to it. Those squares control the types of visualization that appear on the slide. When the presenter presses the square, the visualization changes. Figure 3.17 shows the different types of visualizations available. The left chart is a bar plot and corresponds to the red square with the number 1. The middle chart is a scatter plot and matches the red square with the number 2. The chart on the right is a line plot and corresponds with the red square with the number 3.

The second icon (plus symbol inside a circle), when hovered, produces the two inputs and the button next to it. Those input fields accept numbers that correspond to the aforementioned dataset, and when the user presses the ADD button, it will change the current dataset and update the live visualization. Furthermore, to remove an entry/point for the dataset, all the presenter has to do is press on the corresponding bar/point to that entry.


Alt Text

How would you describe this object and its context to someone who is blind?

(1-2 sentences recommended)

```
%%liveviz("sample.csv")%%
```

☐ Mark as decorative

Figure 3.15: Live visualization plugin features in Microsoft's Powerpoint (top figure) and the respective translation in the new Dynaslides format (bottom figure).

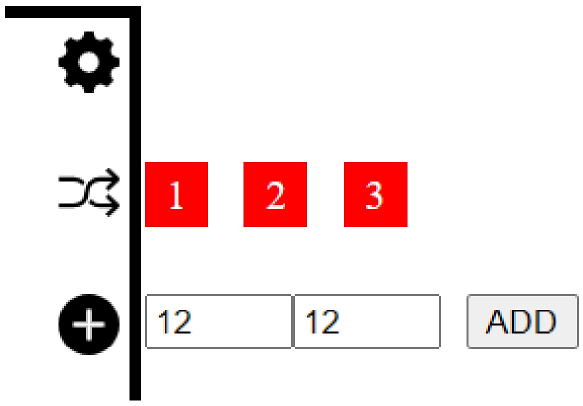


Figure 3.16: Live visualization plugin settings menu.

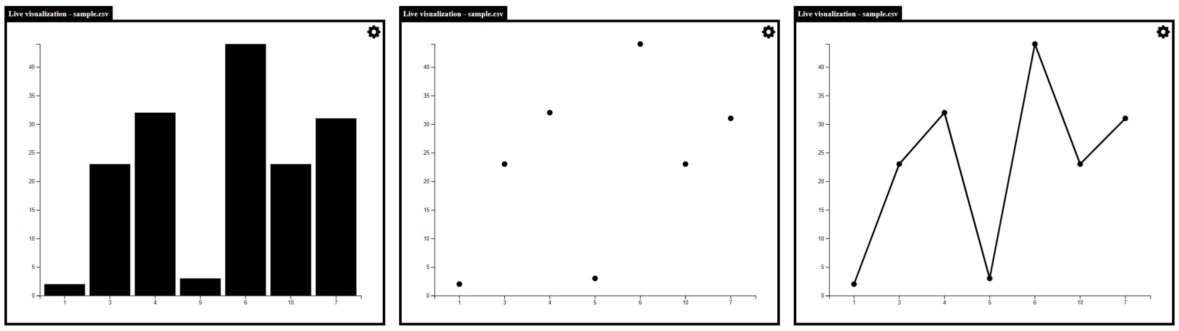


Figure 3.17: Difference available visualizations for the dataset provided in the live visualization plugin.

All the aforementioned features are possible due to **d3.js** Javascript library, which allows the plugin to manipulate documents based on data. It helps to bring data to life using HTML, SVG, and CSS. The live visualization plugin has three different visualizations stored and outputs the one corresponding to the presenter's chosen visualization.

Finally, the visualization's customization, i.e., colors and transitions, are left for the user to change afterward. We first tried to give the bars/points present in the visualization, the colors from the Microsoft's Powerpoint theme chosen for the corresponding presentation. However, it did not turn out to be a good feature since themes may have a variable number of colors, and attributing random colors to the dataset would not match the slides in most cases.

3.6.4 Live Audience Interaction

This plugin offers a live question functionality to interact with the audience. It allows the presenter to create a yes or no question to a given audience, where they can consequentially vote on their own devices and see the results in the respective slide.

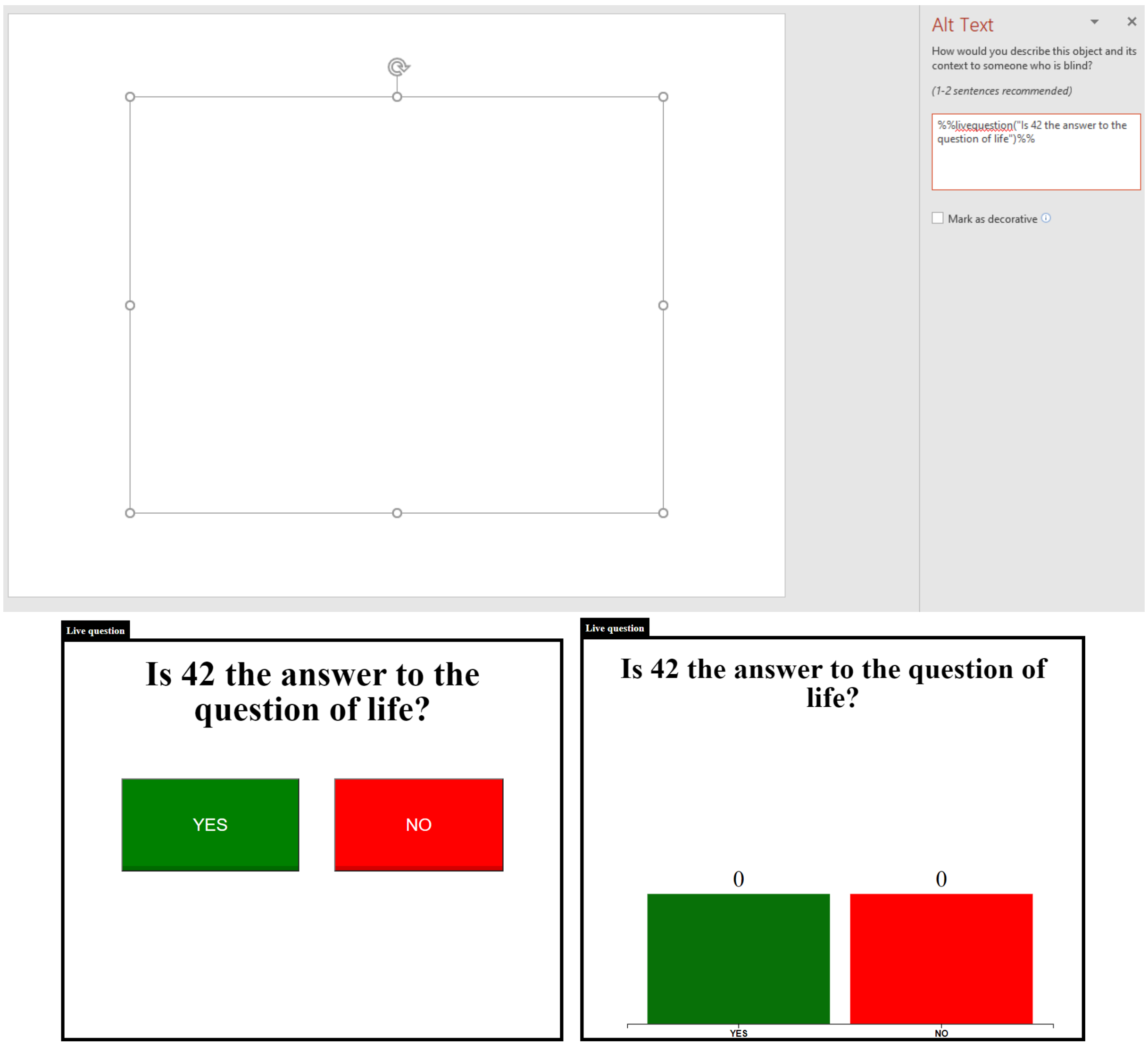
To do this, it's first important to state that this plugin will not function without activated audience feedback. This occurs because the plugin needs to access both views offered by the aforementioned audience feedback system to function correctly. The audience needs to have access to the client presentation, where they can vote and then watch the results in the master presentation.

In this case, the master presentation will transform the text box from the original presentation into a bar plot with two bars and with the respective question. One shows the count for the number of positive answers, and the other shows the count for the negative answers. This is done using the aforementioned **d3.js** Javascript library, which is the main library of the live visualization plugin.

The client presentation will use the text box from the original presentation to create a set of HTML buttons in conjunction with the respective question, where one casts a vote for a positive answer and another for a negative answer. The communication system between the presentations is the same as the one used by the audience feedback shared plugin functionality.

As shown in figure 3.18, the plugin uses the tag `livequestion` and the option that comes with it, which in this case is the yes or no question the presenter wants to ask. It will use the text box's position and size to generate a visualization for the presenter view and use those dimensions to create the two aforementioned buttons for the voting system to work. The visualization on the presenter view will update according to each vote. After viewers vote on their device, it will display a message saying the user should check the results on the master presentation.

Figure 3.19 shows the process of one user voting positively on the displayed question. The top left slide is the client view, and the top right is the master view before the user votes. After the user votes yes on the top left slide, both views transform into the bottom slides. The bottom left slide shows a message thanking the user for voting and advises the user to look for the global results on the master presentation. The bottom right slide shows the global results, which is the aggregation of data from the users connected to it.

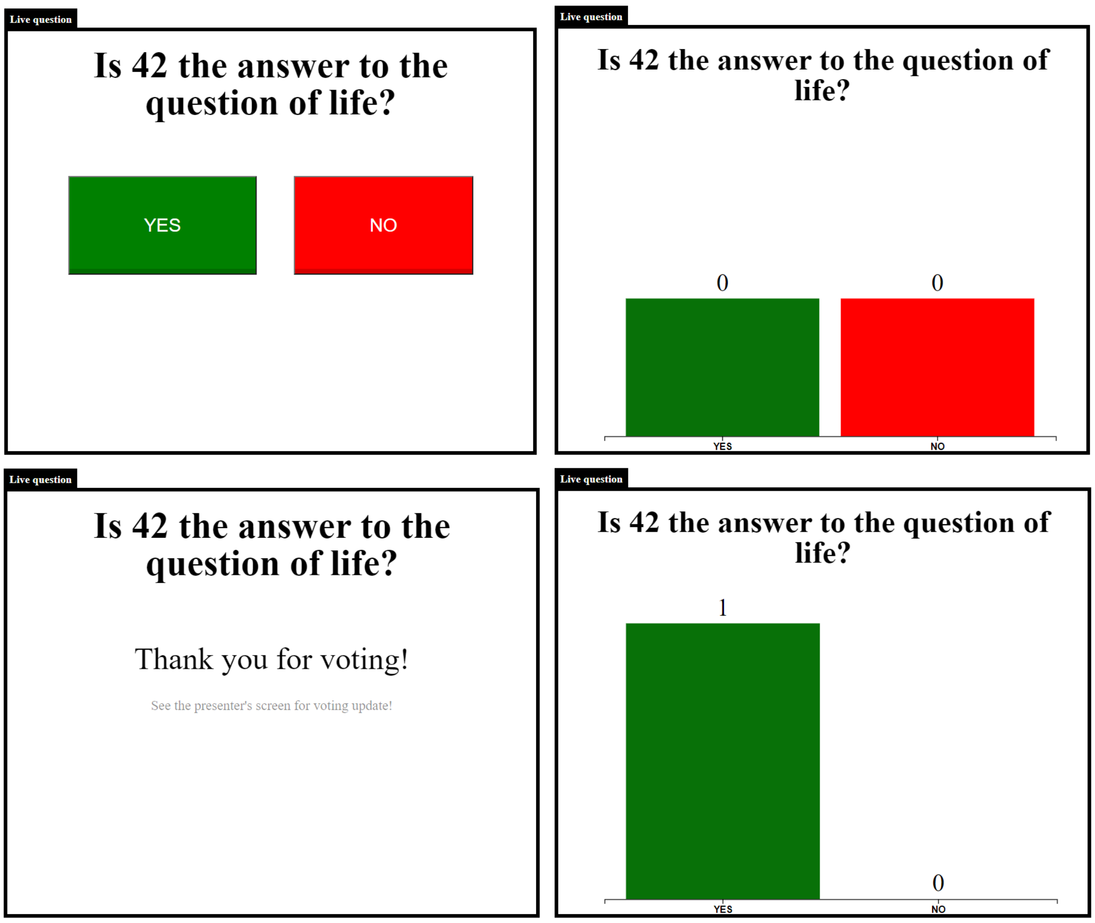

Alt Text
How would you describe this object and its context to someone who is blind?
(1-2 sentences recommended)
Mark as decorative
Live question
YES
NO
Live question
0
0
YES
NO

```
%%livequestion("Is 42 the answer to the question of life")%%
```

Is 42 the answer to the question of life?

Is 42 the answer to the question of life?

Figure 3.18: Live audience plugin features in Microsoft's Powerpoint (top figure) and the respective translation in the new Dynaslides format (bottom figure).


Live question

Is 42 the answer to the question of life?

Live question

Is 42 the answer to the question of life?

Live question

Is 42 the answer to the question of life?

Thank you for voting!

See the presenter's screen for voting update!

Live question

Is 42 the answer to the question of life?

Figure 3.19: Live audience plugin differences between the two views before the connected user voted (top two figures) and the respective state after the user voted yes (bottom two figures).

4 Evaluation

Contents

In this chapter, we opted for two methodologies to properly evaluate the application developed, each with a different purpose.

Firstly, we chose to execute a custom set of predefined tests composed of several Microsoft's Powerpoint presentations. Each one has a certain type of feature in it. We want to ensure the translated document maintained a high level of fidelity compared to the original document, meaning that the new dynaslides presentation (Reveal.js/HTML) bears almost the same appearance as the original presentation.

Secondly, we conducted a series of usability tests with a sample population to understand if the software developed fulfills the requirements we wanted to achieve.

In the following sections, we will also discuss the results obtained from each methodology and the population sample used for the usability evaluation.

4.1 Translation Scrutiny Methodology

To ensure the translation to the new dynaslides format (HTML5 with reveal.js) meets the original presentation standards as much as possible, we devised a process to guarantee that most of the features in the Microsoft's Powerpoint presentations are present in the new format. The process consists of Microsoft's Powerpoint presentations, divided by groups of different features that we translate when new changes to the code or process are executed. Namely:

1. Position - This set ensures the items present in the new format are almost the same as in the original presentation (Position, size, and rotation).
2. Image and Effects - This set ensures the images and their effects are correctly translated in the new format.
3. Shapes - This set ensures that a set of basic shapes are translated correctly to the new format. The shapes are already mentioned in the solution. We couldn't cover them all due to the large number of shapes offered by Microsoft's Powerpoint.
4. Color - This set ensures the color and consequential color effects are correctly translated in the new format.
5. Themes - This set ensures that the application can translate a couple of themes with different characteristics between them.
6. Complex Tests - This set executes through a complex set of Microsoft's PowerPoint presentations that have all the elements mentioned above plus having a large file size, thus ensuring the translation ends in an acceptable amount on time.

4.2 User Population

In this set of usability tests, we asked 24 volunteers to execute several tasks, each one targeting a particular feature in the process of translating a Microsoft's Powerpoint presentation with or without an interactive plugin.

4.2.1 User Profile

The volunteers range from 20 to 24 years old, the majority being 23 years old. They also have education levels ranging from primary school to master's degrees, the most recurring being the latter. Another important factor to consider in the discussion ahead is that most volunteers possess education in Engineering and Related Technologies, and Information Technology, although some volunteers stated some other fieldsA.3.

We also asked the volunteers their level of knowledge in Microsoft's Powerpoint and how often they used the software. We wanted to make sure that at least the users knew how to use the basic functions of said software, or else it wouldn't make sense to participate in our set of tasks. The level of knowledge was based on a table [1] provided by the Canadian University of Concordia. All the relevant data is shown in A.

Although our test group is restrained to a specific age range, our goal here was to evaluate a tool targeted towards lecture-based, Microsoft's Powerpoint enthusiastic environments. Hence, we wouldn't be able to make good assumptions with other age/skill groups.

4.3 Usabilty Evaluation Methodology

We started by giving a quick explanation of what the test was about, the purpose of the application developed, and what was expected from the volunteers in the tests. Since we currently live with an unfortunate Covid-19 pandemic, all the tests were conducted using an online streaming tool that enabled the volunteer to control the test environment in the individual's computer controlling the test.

The volunteers were then asked to answer the first section of the user questionnaire. We gathered the information stated in A.5. Also, we asked if they agreed with the recording of the test for further analysis.

[1]https://www.concordia.ca/content/dam/concordia/services/hr/docs/employment/guides/proficiency-computer-skills.pdf

4.3.1 Environment

After answering the questionnaire, we explained to the volunteers the tools needed for the test (already sharing the test controller's computer) screen. We told them that they could only use the Dynaslides website version, Microsoft's Powerpoint, and the reference guide explaining how to use the website and the plugins we developed. We then proceeded to a quick demonstration of creating a plugin in Microsoft's Powerpoint, translating that file, and then opening it in the browser. Finally, we gave each user a maximum of five minutes to experiment with the process's three aforementioned parts.

4.3.2 Tasks

We conceived nine different tasks, each one focused on a particular functionality or path that our application demands. The tasks were the following:

1. Use the Dynaslides application to translate a predefined Powerpoint file;
2. Create a new Powerpoint file where you use the live coding plugin with a predefined piece of code and translate it;
3. Open a Dynaslides presentation that has a live coding plugin example and run the code that appears on the first slide;
4. Create a new Powerpoint file where you use the live visualization plugin with a predefined data document and translate it;
5. Open a Dynaslides presentation that has a live visualization plugin example and change between the available charts, add a new point and remove it;
6. Use the Dynaslides application to translate a predefined Powerpoint file with the audience feedback activated;
7. Open two Dynaslides presentations (one corresponding to the audience and another to the presenter) that have the audience feedback activated and give a ”like” to the first slides, a ”dislike” to the second slides and send a message in the first slide. Finally, check if the presenter got the correct data;
8. Create a new Powerpoint file where you use the live audience interaction plugin with a predefined question document and translate it;
9. Open two Dynaslides presentations (one corresponding to the audience and another to the presenter) that have the audience feedback activated with a live question and vote ”yes” on the prompt available in the slide. After that, check if the bar chart was update in the presenter presentation;

The tests always started with task number 1. This decision guaranteed the basic functionality of translating and opening a dynaslides presentation for the other tasks. The remaining tasks were grouped randomly for each subject so that each task's metrics were independent of the order they were performed.

During the tasks, the volunteers were asked to speak what they were thinking, to understand their thought process and emotions. At the end of each task, they were asked to rate the task's difficulty on a scale of 1 to 5, where 1 meaning it was complicated and 5, meaning it was effortless. They were also encouraged to give feedback either at the end of each task or in the questionnaire given at the end of the test.

Furthermore, in each task, we measured the time it took to finish the task, the number of errors performed during the task, and if they were able to complete the task to understand and analyze the efficiency, effectiveness, and completeness of the set of tasks.

4.4 Results

The following results capture both analyses of the methodologies presented before and explain if the features developed for this project are well implemented and easy to use.

4.4.1 Translation

The process mentioned above regarding the translation tests guarantees that the features in the predefined presentations are almost identical in the new format. We will further discuss the limitations of the translation in another chapter. However, we want to emphasize that the features we chose to be present in these sets of tests are the ones we believe are the most used in regular use of the Microsoft's Powerpoint application. This provides the application's standard features and leaves space for further improvement if the presenter requires more complex features.

4.4.2 Usability

We are interested in finding out if the produced application is easy to use while attempting the solve the target problem of this project. Therefore, we will discuss the results by analyzing the tasks by the feature they were meant to test. We also decided to group the most relevant data statistics on the figure 4.1 and make a time per task plot4.2, since it makes it easier for us to relate that data with the consequential analysis.

Firstly, we wanted to test if the process of translating a Microsoft's Powerpoint presentation into a dynaslides presentation was straightforward, and the results show exactly that. It took the volunteers around one minute to translate and open a dynaslides presentation. The number of errors was meager. Although our population size is not substantial, the lower and upper confidence levels show a minor gap. There are also no outliers present in figure 4.2.

This shows that people understood the process well from the demonstration and that the website is quite easy to grasp.

Moreover, we focused on two different sides of the process. Creating a Microsoft's Powerpoint presentation with one of the plugins and using the presentation's plugins while in the browser.

Creating a Microsoft's Powerpoint presentation with a plugin, represented by tasks two, four, and eight, appeared to be a difficult process to grasp for some participants since it required the reference guide. We concluded that some features explain in the reference guide were not coherent.

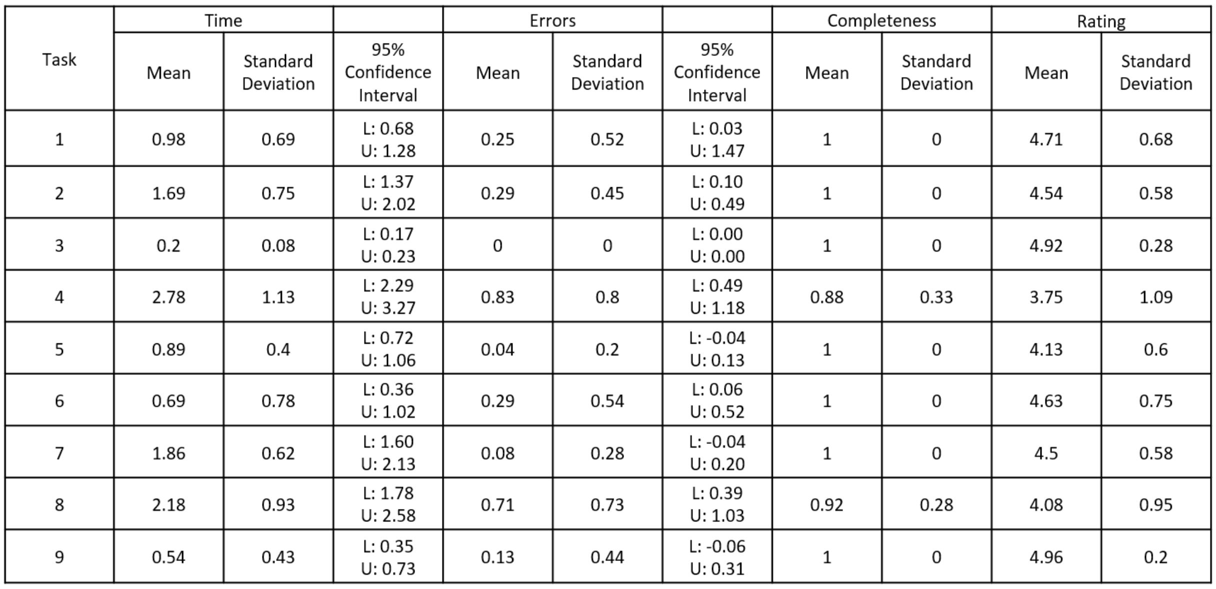
We want to focus on creating the live visualization plugin (Task 4) since it produced the most interesting results of the three plugins. It took the most time to create out of all the three plugins. The task was also the only one that showed a normal distribution according to the Shapiro-Wilk test we performed, seen in appendix AA.

This happened because the plugin required inserting the respective data file's name in the command given in the alt text option. The application needs to have the name of the data file to find the respective file while translating. The reference guide explained how to create this plugin, although some participants didn't read it carefully enough.

On the other hand, another important subject was the fact that it was also required to insert the data file on the website before translating the file. Although the reference guide also explained this, some volunteers complained that there was no feedback when the file was inserted, which created confusion and, consequently, added time to the clock.

Creating a live question also presented high execution times and errors (Task 8) because the user needs to activate the audience feedback before translating the file. Some volunteers couldn't find the reason, which led to complaints regarding not receiving feedback about the errors when translating the file. We can also see this in Task 6. The volunteers had to activate the audience feedback plugin before uploading the original presentation. Although the task was easy to perform, we can see in figure 4.2 two outliers, which means some users took some time and errors to understand the requirement of activating the audience feedback button.

The process of using the browser's plugins, corresponding to tasks three, seven, and nine, went well overall, with most participants understanding how each plugin worked. Task 9 has some outliers, and this happened because some volunteers got this task as their first one after performing Task 1. Since Task 9 corresponds to using the audience interaction plugin, the users had to learn how to open both presentations, vote, and finally observe the results. This led to some volunteers getting confused and taking more time to finish the task.



Task	Time			Errors			Completeness		Rating	
	Mean	Standard Deviation	95% Confidence Interval	Mean	Standard Deviation	95% Confidence Interval	Mean	Standard Deviation	Mean	Standard Deviation
1	0.98	0.69	L: 0.68 U: 1.28	0.25	0.52	L: 0.03 U: 1.47	1	0	4.71	0.68
2	1.69	0.75	L: 1.37 U: 2.02	0.29	0.45	L: 0.10 U: 0.49	1	0	4.54	0.58
3	0.2	0.08	L: 0.17 U: 0.23	0	0	L: 0.00 U: 0.00	1	0	4.92	0.28
4	2.78	1.13	L: 2.29 U: 3.27	0.83	0.8	L: 0.49 U: 1.18	0.88	0.33	3.75	1.09
5	0.89	0.4	L: 0.72 U: 1.06	0.04	0.2	L: -0.04 U: 0.13	1	0	4.13	0.6
6	0.69	0.78	L: 0.36 U: 1.02	0.29	0.54	L: 0.06 U: 0.52	1	0	4.63	0.75
7	1.86	0.62	L: 1.60 U: 2.13	0.08	0.28	L: -0.04 U: 0.20	1	0	4.5	0.58
8	2.18	0.93	L: 1.78 U: 2.58	0.71	0.73	L: 0.39 U: 1.03	0.92	0.28	4.08	0.95
9	0.54	0.43	L: 0.35 U: 0.73	0.13	0.44	L: -0.06 U: 0.31	1	0	4.96	0.2

Figure 4.1: Usability data statistics

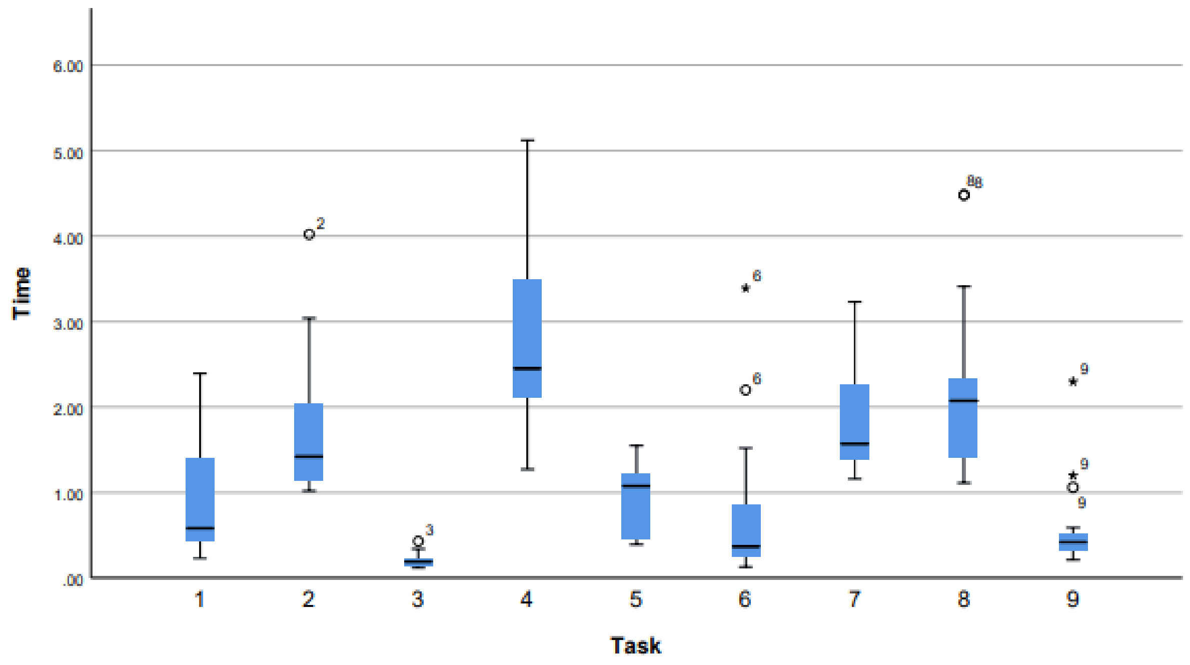


Figure 4.2: Time per task box and whisker plot

Besides that, since the tasks took a small amount of time and had almost no errors, we consider that they were well implemented and intuitive.

4.4.3 Questionnaire

After the tests, we first asked the volunteers three simple yes or no questions, which are the following: ”Do you agree that the use of the alt text feature in the shape format (to activate the plugin) was a good choice?”, ”Do you think this tool would be relevant in an academic environment, such as a lecture?” and ”Do you think this tool would be relevant outside the academic environment?”. The results were 100% yes, 100% yes, and 95.6% yes, respectively. The first question does not add much besides knowing the option we chose works as intended. Since most volunteers don’t know another option to execute this particular functionality, they could not give an insightful opinion. However, with the second and final question, we can say that most volunteers agreed on the fact that the tool is useful in an academic environment and outside of one as well.

The following System Usability Scale (SUS) scored 82 out of 100. Although this questionnaire is a subjective view of the overall system’s usability, we find it very useful and tell us the application implemented the features as intended.

After that, we have the NASA Task Load Index questionnaire, which scored 13.82 out of 100. This is also an excellent score overall. Therefore, we can say that the tests were very effective in measuring if the application implemented the final features correctly.

Finally, we received much feedback at the end of the questionnaire regarding the notification system, i.e., the lack of notifications in several features. One of the most talked-about was that the audience feedback system needed to be activated before translating the original presentation. Although the reference guide mentioned, some volunteers felt like the application should warn them about not activating the audience feedback setting.

Another feature discussed during the feedback was that the input field for the Microsoft’s Powerpoint presentation and the live visualization plugin’s data files did not offer any tooltips or notifications to know if the application got the files correctly.

5

Conclusions

Contents

5.1 Final Remarks

Today's slideware applications are based on old techniques. They do not take advantage of all the available resources during a presentation. The presenter should interact with the target audience, access a dynamic environment while being in the presentation view, and improve each lecture's experience based on feedback as standard available mechanisms.

Therefore, we developed DynaSlides, an application offering a solution that undertakes the aforementioned issues by creating a flexible plugin-based structure, built using a strong foundation and with the help of a transition mechanism from the standard presentation market leader.

We learned about how Microsoft's Powerpoint works and what must be done to convert the available information into a web version. Since they provide an open format, the only thing required for most presentation elements is to translate them from XML to HTML and CSS elements. CSS already provides most of the properties, and it keeps being updated every year. Only some image and color effects required the use of more complex functions to translate the correct effect.

We also learned that the Javascript plugin structure's possibilities are practically endless, and with the available packages today, there can be more complex and interactive features that can rival the market leader.

We believe the application developed in this project is a step forward in the right direction for an interactive presentation environment. Although it can still be improved and it does not provide some functionality the Microsoft's Powerpoint has to offer, the Dynaslides bundle serves as a design and, ultimately, a proof of concept of the need for such an interactive presentation environment and shows that interactivity is achievable through the technology we have available today.

Furthermore, the successful usability tests we executed prove the simplicity and need of the Dynaslides application. With great confidence, we can say that the current market for presentation technology still has much to be improved.

5.2 Limitations and Future Work

Due to the size and complexity of a presentation application such as Microsoft's Powerpoint, our developed presentation environment focused only on the functionality we believe is the core for every presentation technology. This means we only addressed the problems we thought were important to maintain in the new format, and even those are not perfect.

There are still improvements to be made in the translated objects' position. The application can translate only some basic shapes, color effects, and image effects correctly to the new format. Due to the different compositions of the background themes available in Microsoft's Powerpoint, we can only guarantee the correct translation for backgrounds made with certain images and color properties.

There is also the question of efficiency of the translation since we were not focused on the time the developed application takes to translate a presentation. We did not test for this since our goal was to prove a concept for a presentation environment with easy and reliable authoring and live interactive functionality.

There are many possibilities to pursue in this project. The translation still is far from a complete transformation from Microsoft's Powerpoint. Elements, such as slide transitions, animations, videos, and more, can still be implemented with the current system in place.

Furthermore, with the current plugin architecture, a new plugin can easily be implemented, and thus adding a new feature becomes much more accessible. The currently implemented plugins also have room for improvement. Since they are proofs of concept, they can still have a lot of different new features implemented. For example, the code execution in the live coding plugin could be paired with a visualization to explain some algorithms better. The live visualization plugin could have a better menu implemented, where the presenter can change between more idioms and even change the colors used during the presentation. The live question plugin could have more types of questions and answers. The presenter could have another window where the slides' rating and messages would appear during the presentation.

Finally, a presentation could have a separate application that would allow a presenter to view the state of the live presentation after it was over. This would give the presenter the power to better change the presentation based on facts at a given point in time.

The possibilities with this project are endless and could lead to a much better presentation environment over time.

Bibliography

[1] D. Edge, J. Savage, and K. Yatani, "Hyperslides: dynamic presentation prototyping," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2013, pp. 671–680.

[2] R. P. Spicer, Y.-R. Lin, and A. Kelliher, "Nextslideplease: Agile hyperpresentations," in *Proceedings of the 17th ACM International Conference on Multimedia*, ser. MM '09. New York, NY, USA: ACM, 2009, pp. 1045–1048. [Online]. Available: http://doi.acm.org/10.1145/1631272.1631508

[3] R. Roels and B. Signer, "Mindxpres: An extensible content-driven cross-media presentation platform," in *International Conference on Web Information Systems Engineering*. Springer, 2014, pp. 215–230.

[4] R. Roels, P. Meştereagă, and B. Signer, "An interactive source code visualisation plug-in for the mindxpres presentation platform," in *International Conference on Computer Supported Education*. Springer, 2015, pp. 169–188.

[5] J. Lanir, K. S. Booth, and A. Tang, "Multipresenter: a presentation system for (very) large display surfaces," in *Proceedings of the 16th ACM international conference on Multimedia*. ACM, 2008, pp. 519–528.

[6] M. Thorogood, "slidedeck. js: A platform for generating accessible and interactive web-based course content," in *Proceedings of the 21st Western Canadian Conference on Computing Education*. ACM, 2016, p. 13.

[7] R. Roels and B. Signer, "A conceptual framework and content model for next generation presentation solutions," *Proc. ACM Hum.-Comput. Interact.*, vol. 3, no. EICS, pp. 7:1–7:22, Jun. 2019. [Online]. Available: http://doi.acm.org/10.1145/3331149

[8] R. Roels, Y. Baeten, and B. Signer, "An interactive data visualisation approach for next generation presentation tools," in *Proceedings of the 8th International Conference on Computer Supported Education*. SCITEPRESS-Science and Technology Publications, Lda, 2016, pp. 123–133.

[9] R. J. Craig and J. H. Amernic, “Powerpoint presentation technology and the dynamics of teaching,” *Innovative Higher Education*, vol. 31, no. 3, pp. 147–160, 2006.

[10] I. Parker, “Absolute powerpoint,” *The New Yorker*, vol. 28, pp. 76–87, 2001.

[11] G. Shaw, R. Brown, and P. Bromiley, “Strategic stories: How 3m is rewriting business planning,” *Harvard business review*, vol. 76, no. 3, pp. 41–49, 1998.

[12] E. R. Tufte, *Beautiful evidence*. Graphics Press Cheshire, CT, 2006, vol. 1.

[13] D. Cyphert, “The problem of powerpoint: Visual aid or visual rhetoric?” *Business Communication Quarterly*, vol. 67, no. 1, pp. 80–84, 2004.

[14] L. Good and B. B. Bederson, “Zoomable user interfaces as a medium for slide show presentations,” *Information Visualization*, vol. 1, no. 1, pp. 35–49, 2002.

[15] P. Albinson, “Enhancing programming lectures using interactive web-based lecture slides,” 2016.

[16] A. Holzinger, M. Kickmeier, and D. Albert, “Dynamic media in computer science education; content complexity and learning performance: Is less more?” *Educational Technology Society*, vol. 11, pp. 279–290, 01 2008.

[17] C. E. George, “Experiences with novices: The importance of graphical representations in supporting mental mode.” in *PPIG*, 2000, p. 3.

[18] Y. Sun, D. Lambert, M. Uchida, and N. Remy, “Collaboration in the cloud at google,” in *Proceedings of the 2014 ACM Conference on Web Science*, ser. WebSci ’14. New York, NY, USA: ACM, 2014, pp. 239–240. [Online]. Available: http://doi.acm.org/10.1145/2615569.2615637

[19] C. N. Klokmose, J. R. Eagan, S. Baader, W. Mackay, and M. Beaudouin-Lafon, “Webstrates: Shareable dynamic media,” in *Proceedings of the 28th Annual ACM Symposium on User Interface Software &#38; Technology*, ser. UIST ’15. New York, NY, USA: ACM, 2015, pp. 280–290. [Online]. Available: http://doi.acm.org/10.1145/2807442.2807446

A

Usability Tests Information

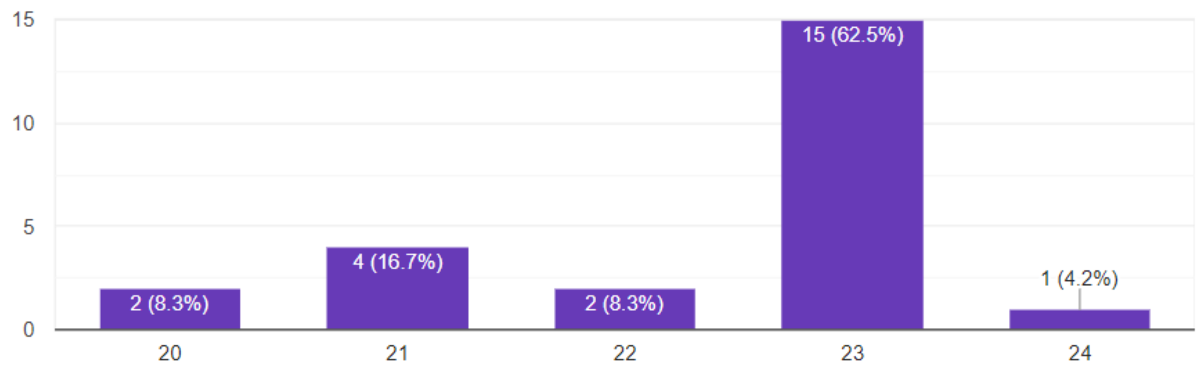


Figure A.1: Age data obtained before usability evaluation

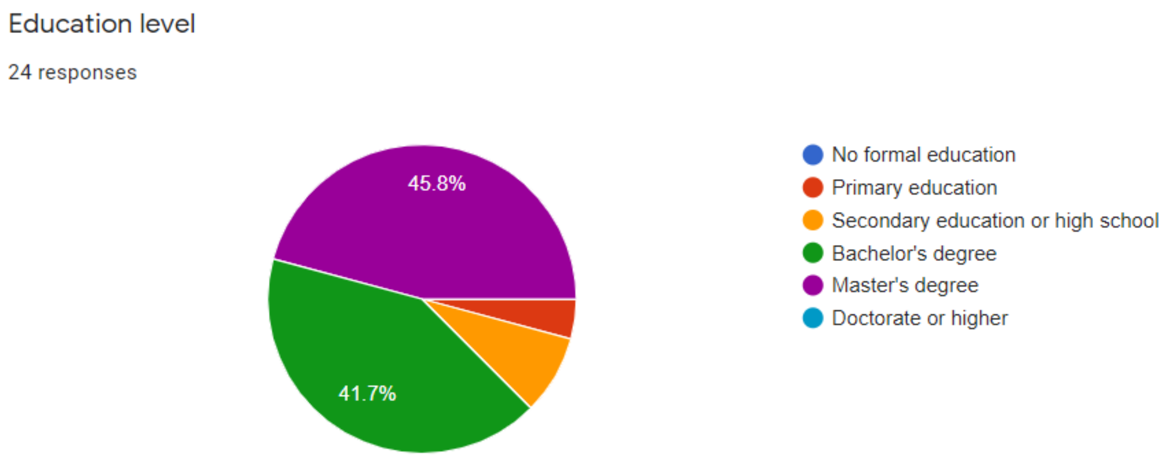


Figure A.2: Education level data obtained before usability evaluation

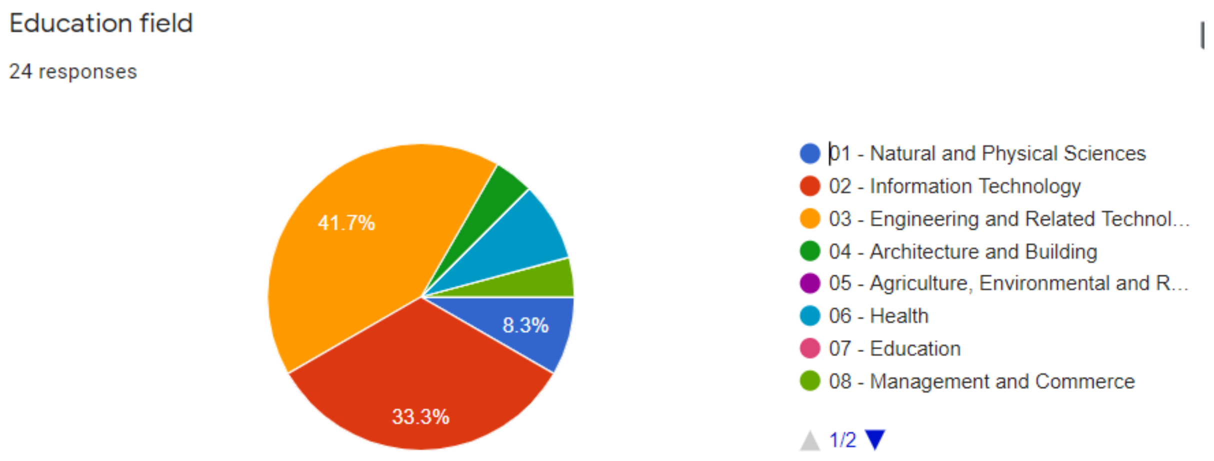


Figure A.3: Education field data obtained before usability evaluation

Descriptives

	id			Statistic	Std. Error
time	1	Mean		.9796	.14291
		95% Confidence Interval for Mean	Lower Bound	.6839	
			Upper Bound	1.2752	
		5% Trimmed Mean		.9434	
		Median		.5800	
		Variance		.490	
		Std. Deviation		.70012	
		Minimum		.23	
		Maximum		2.39	
		Range		2.16	
		Interquartile Range		1.05	
		Skewness		.876	.472
		Kurtosis		-.675	.918
	2	Mean		1.6921	.15668
		95% Confidence Interval for Mean	Lower Bound	1.3680	
			Upper Bound	2.0162	
		5% Trimmed Mean		1.6090	
		Median		1.4200	
		Variance		.589	
		Std. Deviation		.76757	
		Minimum		1.02	
		Maximum		4.02	
		Range		3.00	
		Interquartile Range		.90	
		Skewness		1.686	.472
		Kurtosis		2.606	.918
	3	Mean		.2012	.01593
		95% Confidence Interval for Mean	Lower Bound	.1683	
			Upper Bound	.2342	
		5% Trimmed Mean		.1939	
		Median		.1900	
		Variance		.006	
		Std. Deviation		.07804	
		Minimum		.12	
		Maximum		.43	

Page 1

Descriptives

id			Statistic	Std. Error
	Range		.31	
	Interquartile Range		.09	
	Skewness		1.425	.472
	Kurtosis		2.069	.918
4	Mean		2.7779	.23584
	95% Confidence Interval for Mean	Lower Bound	2.2900	
		Upper Bound	3.2658	
	5% Trimmed Mean		2.7326	
	Median		2.4500	
	Variance		1.335	
	Std. Deviation		1.15538	
	Minimum		1.27	
	Maximum		5.12	
	Range		3.85	
	Interquartile Range		1.43	
	Skewness		.568	.472
	Kurtosis		-.518	.918
5	Mean		.8892	.08373
	95% Confidence Interval for Mean	Lower Bound	.7160	
		Upper Bound	1.0624	
	5% Trimmed Mean		.8803	
	Median		1.0750	
	Variance		.168	
	Std. Deviation		.41021	
	Minimum		.39	
	Maximum		1.55	
	Range		1.16	
	Interquartile Range		.76	
	Skewness		.035	.472
	Kurtosis		-1.713	.918
6	Mean		.6946	.16168
	95% Confidence Interval for Mean	Lower Bound	.3601	
		Upper Bound	1.0290	
	5% Trimmed Mean		.5869	
	Median		.3650	

Page 2

Descriptives

id				Statistic	Std. Error
		Variance		.627	
		Std. Deviation		.79206	
		Minimum		.13	
		Maximum		3.39	
		Range		3.26	
		Interquartile Range		.77	
		Skewness		2.206	.472
		Kurtosis		5.118	.918
	7	Mean		1.8621	.12899
		95% Confidence Interval for Mean	Lower Bound	1.5953	
			Upper Bound	2.1289	
		5% Trimmed Mean		1.8265	
		Median		1.5700	
		Variance		.399	
		Std. Deviation		.63190	
		Minimum		1.16	
		Maximum		3.23	
		Range		2.07	
		Interquartile Range		.89	
		Skewness		.840	.472
		Kurtosis		-.297	.918
	8	Mean		2.1796	.19366
		95% Confidence Interval for Mean	Lower Bound	1.7790	
			Upper Bound	2.5802	
		5% Trimmed Mean		2.1107	
		Median		2.0750	
		Variance		.900	
		Std. Deviation		.94875	
		Minimum		1.11	
		Maximum		4.48	
		Range		3.37	
		Interquartile Range		.99	
		Skewness		1.183	.472
		Kurtosis		1.157	.918
	9	Mean		.5388	.09018

Descriptives

	id			Statistic	Std. Error
		95% Confidence Interval for Mean	Lower Bound	.3522	
			Upper Bound	.7253	
		5% Trimmed Mean		.4691	
		Median		.4200	
		Variance		.195	
		Std. Deviation		.44181	
		Minimum		.21	
		Maximum		2.30	
		Range		2.09	
		Interquartile Range		.21	
		Skewness		3.148	.472
		Kurtosis		11.286	.918
errors	1	Mean		.25	.109
		95% Confidence Interval for Mean	Lower Bound	.03	
			Upper Bound	.47	
		5% Trimmed Mean		.18	
		Median		.00	
		Variance		.283	
		Std. Deviation		.532	
		Minimum		0	
		Maximum		2	
		Range		2	
		Interquartile Range		0	
		Skewness		2.131	.472
		Kurtosis		4.143	.918
	2	Mean		.29	.095
		95% Confidence Interval for Mean	Lower Bound	.10	
			Upper Bound	.49	
		5% Trimmed Mean		.27	
		Median		.00	
		Variance		.216	
		Std. Deviation		.464	
		Minimum		0	
		Maximum		1	
		Range		1	

Descriptives

id			Statistic	Std. Error
	Interquartile Range		1	
	Skewness		.979	.472
	Kurtosis		-1.145	.918
3	Mean		.00	.000
	95% Confidence Interval for Mean	Lower Bound	.00	
		Upper Bound	.00	
	5% Trimmed Mean		.00	
	Median		.00	
	Variance		.000	
	Std. Deviation		.000	
	Minimum		0	
	Maximum		0	
	Range		0	
	Interquartile Range		0	
	Skewness		.	.
	Kurtosis		.	.
4	Mean		.83	.167
	95% Confidence Interval for Mean	Lower Bound	.49	
		Upper Bound	1.18	
	5% Trimmed Mean		.81	
	Median		1.00	
	Variance		.667	
	Std. Deviation		.816	
	Minimum		0	
	Maximum		2	
	Range		2	
	Interquartile Range		2	
	Skewness		.329	.472
	Kurtosis		-1.409	.918
5	Mean		.04	.042
	95% Confidence Interval for Mean	Lower Bound	-.04	
		Upper Bound	.13	
	5% Trimmed Mean		.00	
	Median		.00	
	Variance		.042	

Descriptives

id			Statistic	Std. Error
	Std. Deviation		.204	
	Minimum		0	
	Maximum		1	
	Range		1	
	Interquartile Range		0	
	Skewness		4.899	.472
	Kurtosis		24.000	.918
6	Mean		.29	.112
	95% Confidence Interval for Mean	Lower Bound	.06	
		Upper Bound	.52	
	5% Trimmed Mean		.22	
	Median		.00	
	Variance		.303	
	Std. Deviation		.550	
	Minimum		0	
	Maximum		2	
	Range		2	
	Interquartile Range		1	
	Skewness		1.800	.472
	Kurtosis		2.676	.918
7	Mean		.08	.058
	95% Confidence Interval for Mean	Lower Bound	-.04	
		Upper Bound	.20	
	5% Trimmed Mean		.04	
	Median		.00	
	Variance		.080	
	Std. Deviation		.282	
	Minimum		0	
	Maximum		1	
	Range		1	
	Interquartile Range		0	
	Skewness		3.220	.472
	Kurtosis		9.124	.918
8	Mean		.71	.153

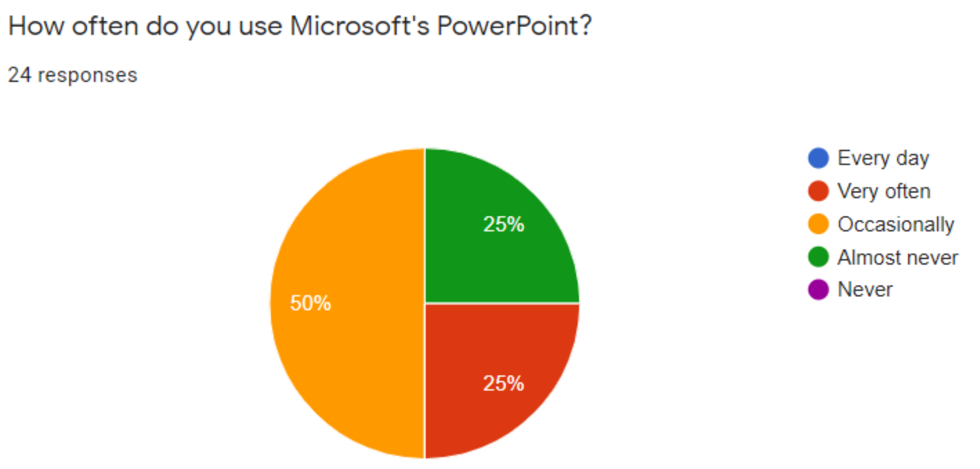
Descriptives

id				Statistic	Std. Error
		95% Confidence Interval for Mean	Lower Bound	.39	
			Upper Bound	1.03	
		5% Trimmed Mean		.68	
		Median		1.00	
		Variance		.563	
		Std. Deviation		.751	
		Minimum		0	
		Maximum		2	
		Range		2	
		Interquartile Range		1	
		Skewness		.553	.472
		Kurtosis		-.950	.918
	9	Mean		.13	.092
		95% Confidence Interval for Mean	Lower Bound	-.06	
			Upper Bound	.31	
		5% Trimmed Mean		.04	
		Median		.00	
		Variance		.201	
		Std. Deviation		.448	
		Minimum		0	
		Maximum		2	
		Range		2	
		Interquartile Range		0	
		Skewness		3.797	.472
		Kurtosis		14.650	.918

Tests of Normality

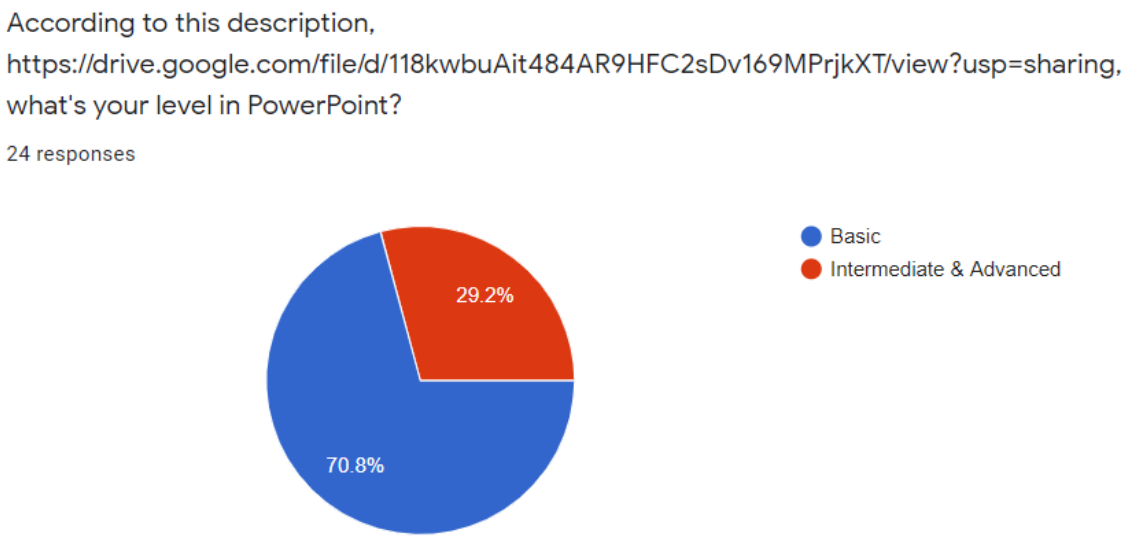
		Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	id	Statistic	df	Sig.	Statistic	df	Sig.
time	1	.253	24	.000	.840	24	.001
	2	.277	24	.000	.793	24	.000
	3	.149	24	.181	.860	24	.003
	4	.158	24	.124	.931	24	.104
	5	.240	24	.001	.845	24	.002
	6	.342	24	.000	.688	24	.000
	7	.214	24	.006	.884	24	.010
	8	.220	24	.004	.861	24	.004
	9	.332	24	.000	.609	24	.000
errors	1	.473	24	.000	.531	24	.000
	2	.443	24	.000	.573	24	.000
	3	.	24	.	.	24	.
	4	.263	24	.000	.789	24	.000
	5	.539	24	.000	.209	24	.000
	6	.452	24	.000	.580	24	.000
	7	.533	24	.000	.316	24	.000
	8	.286	24	.000	.779	24	.000
	9	.526	24	.000	.316	24	.000

a. Lilliefors Significance Correction


How often do you use Microsoft's PowerPoint?

24 responses

Figure A.4: Microsoft's Powerpoint usage data obtained before usability evaluation


According to this description, https://drive.google.com/file/d/118kwbuAit484AR9HFC2sDv169MPrjkXT/view?usp=sharing, what's your level in PowerPoint?

24 responses

Figure A.5: Microsoft's Powerpoint level data obtained before usability evaluation

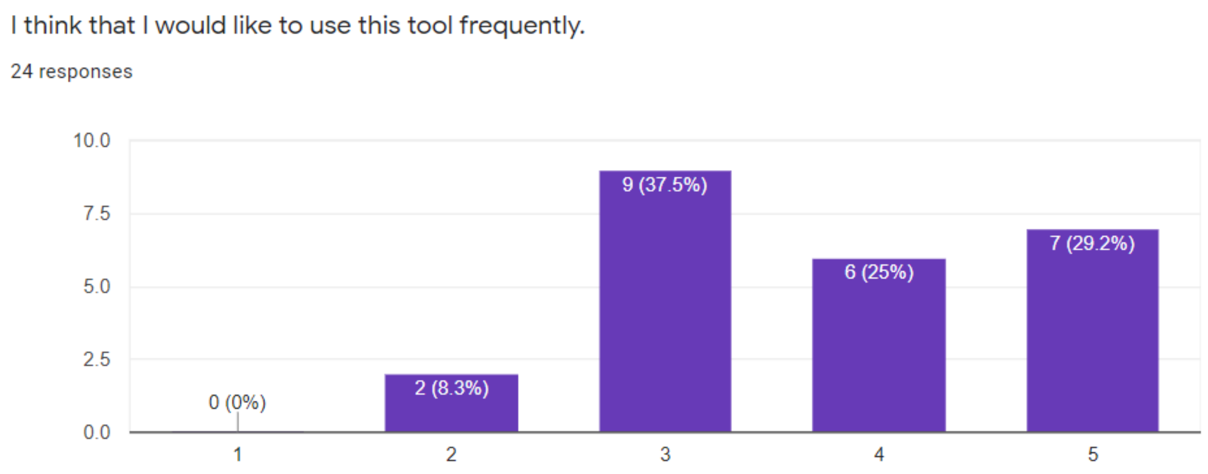


Figure A.6: System Usability Scale (SUS) question number 1

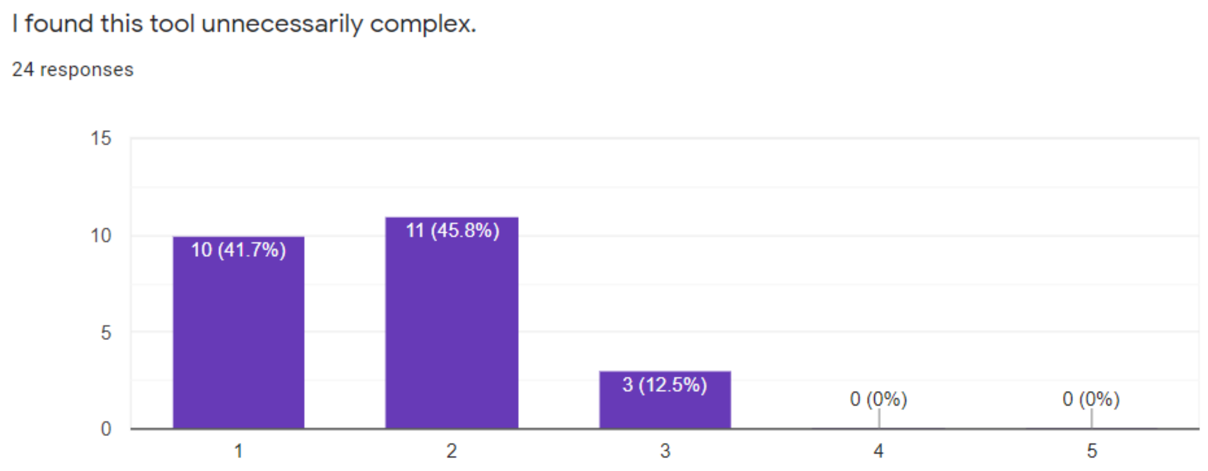


Figure A.7: System Usability Scale (SUS) question number 2

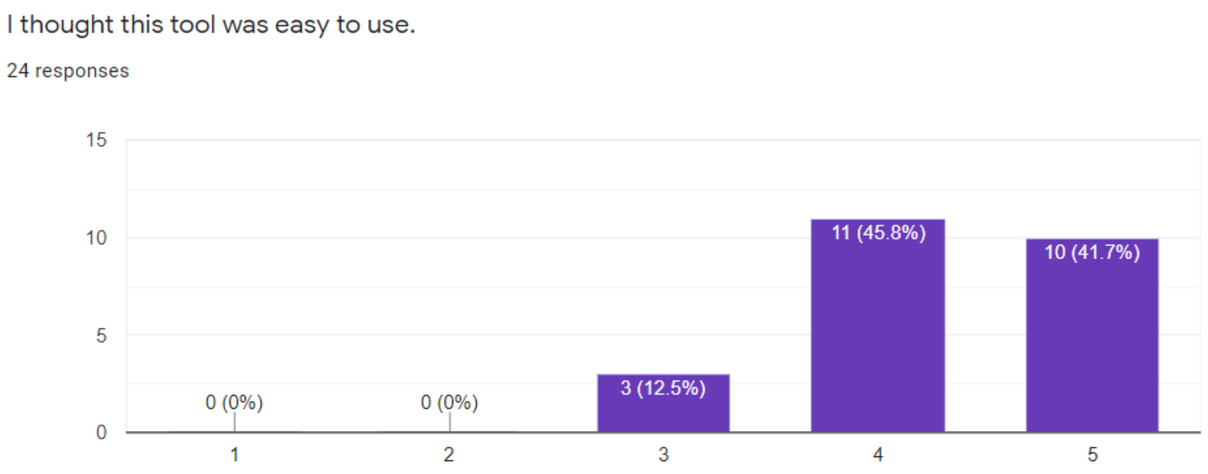


Figure A.8: System Usability Scale (SUS) question number 3

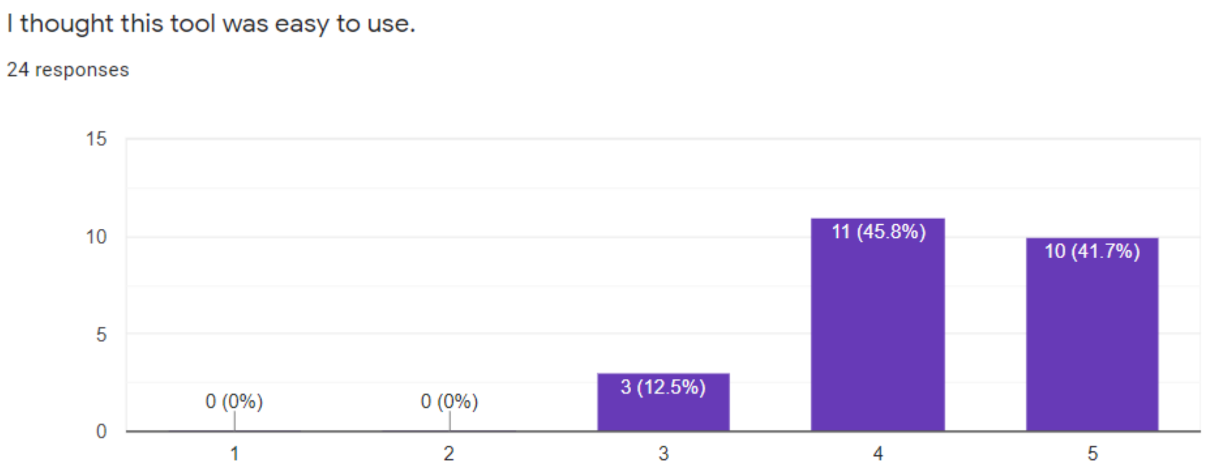


Figure A.9: System Usability Scale (SUS) question number 3

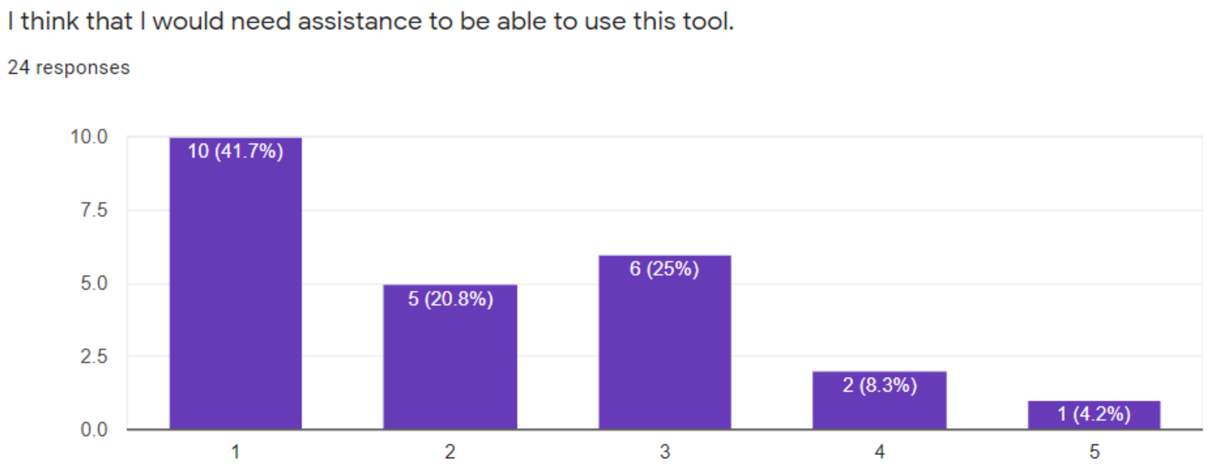


Figure A.10: System Usability Scale (SUS) question number 4

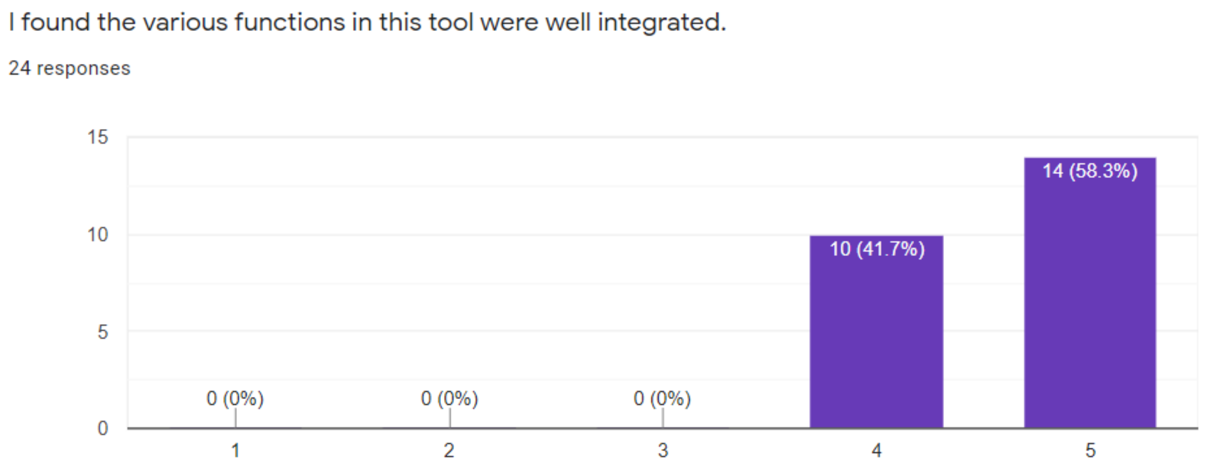


Figure A.11: System Usability Scale (SUS) question number 5

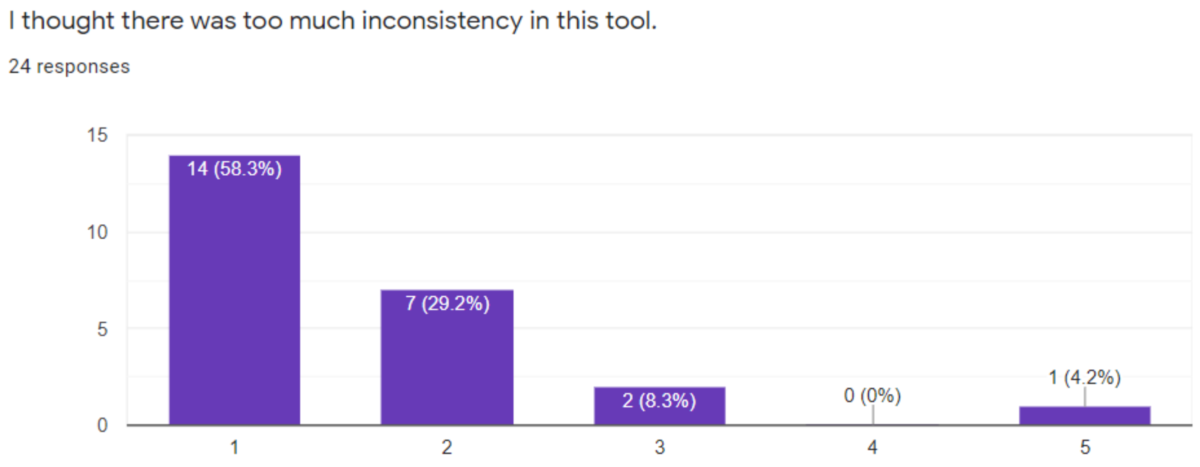


Figure A.12: System Usability Scale (SUS) question number 6

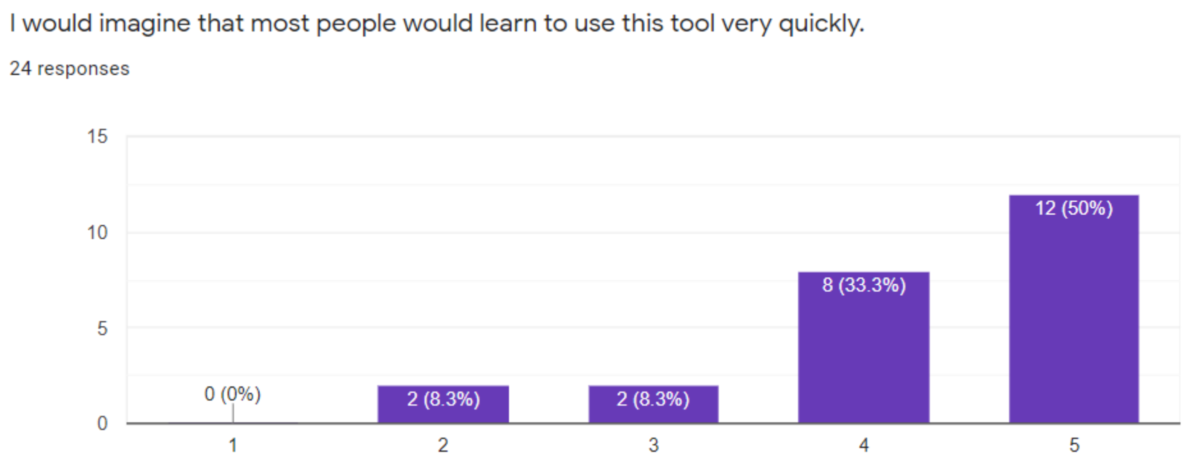


Figure A.13: System Usability Scale (SUS) question number 7

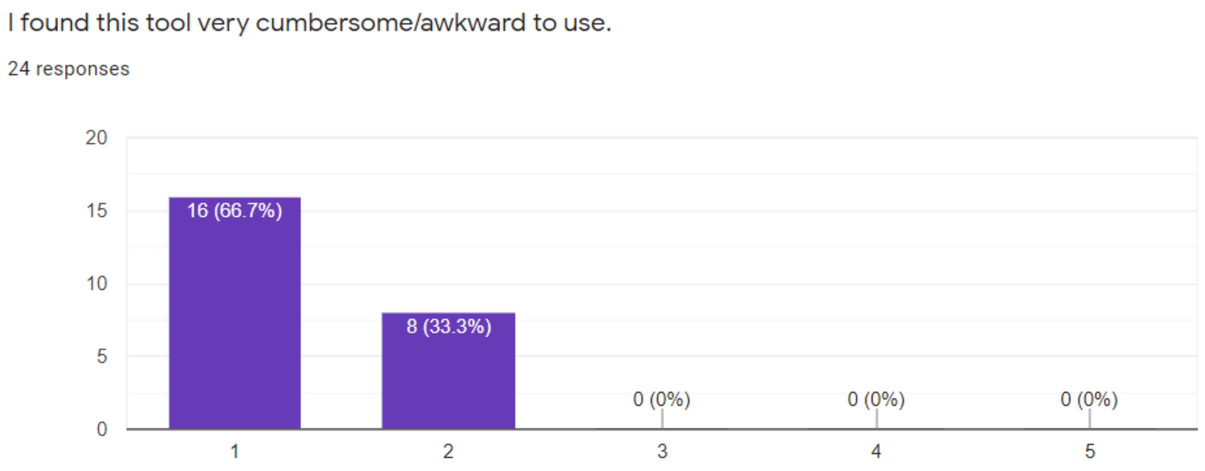


Figure A.14: System Usability Scale (SUS) question number 8

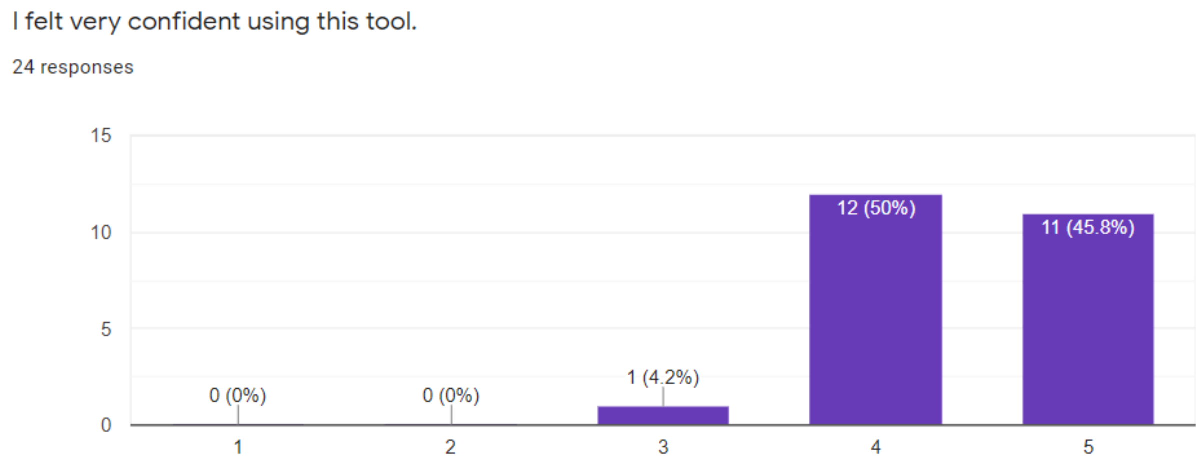


Figure A.15: System Usability Scale (SUS) question number 9

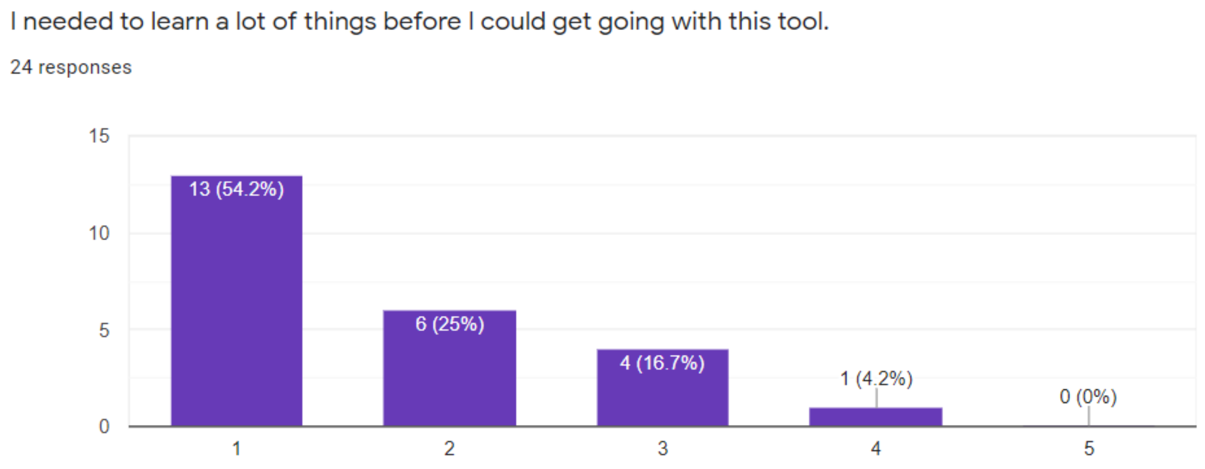


Figure A.16: System Usability Scale (SUS) question number 10

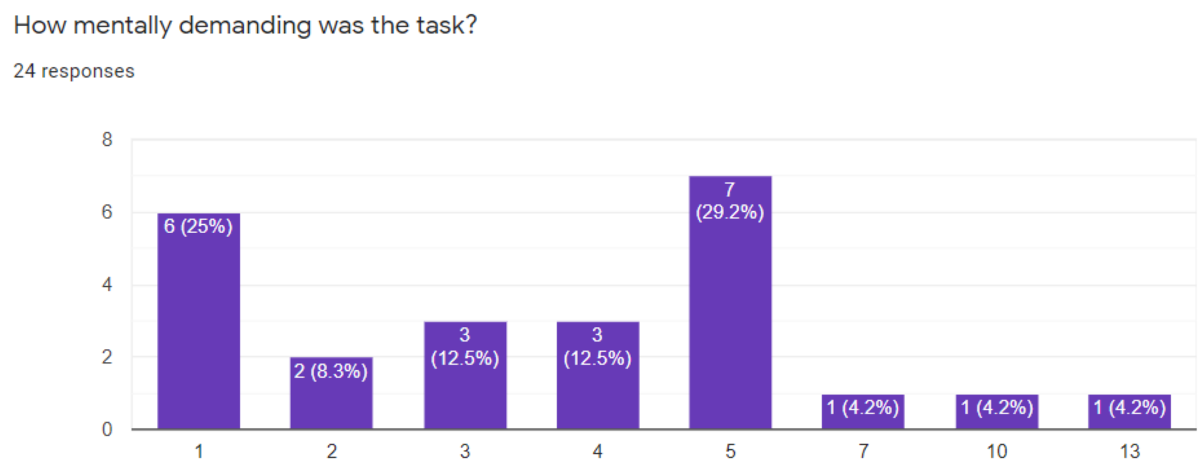


Figure A.17: Nasa TLX index question number 1

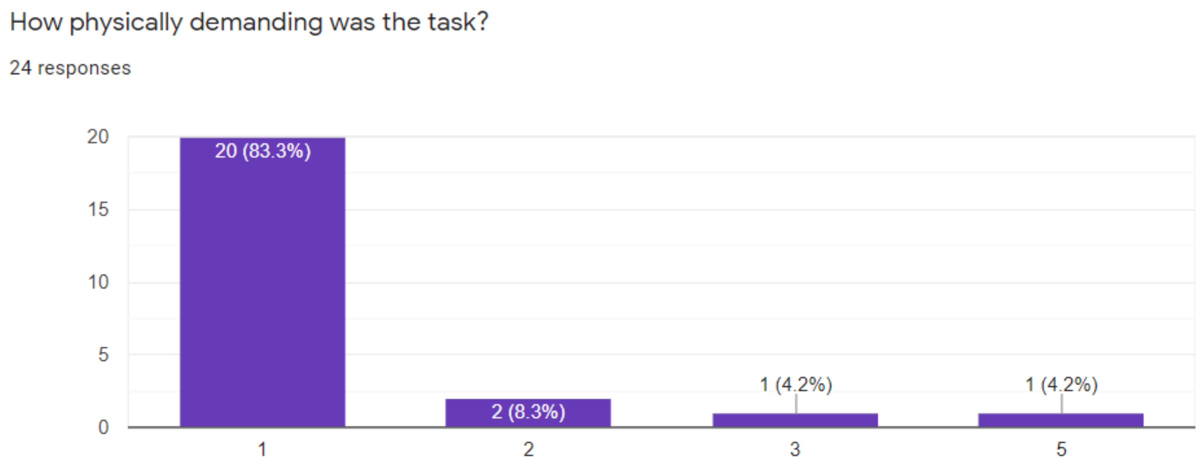


Figure A.18: Nasa TLX index question number 2

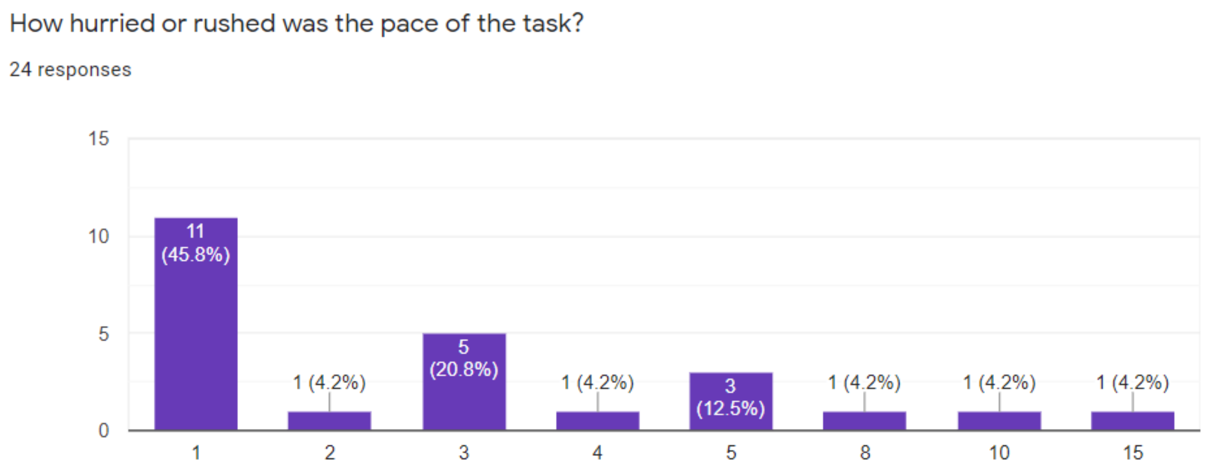


Figure A.19: Nasa TLX index question number 3

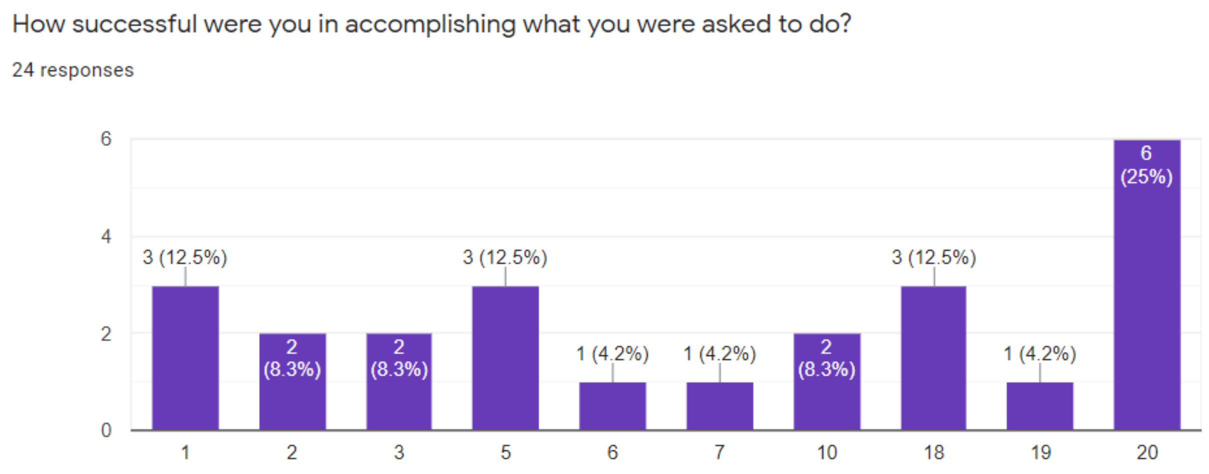


Figure A.20: Nasa TLX index question number 4

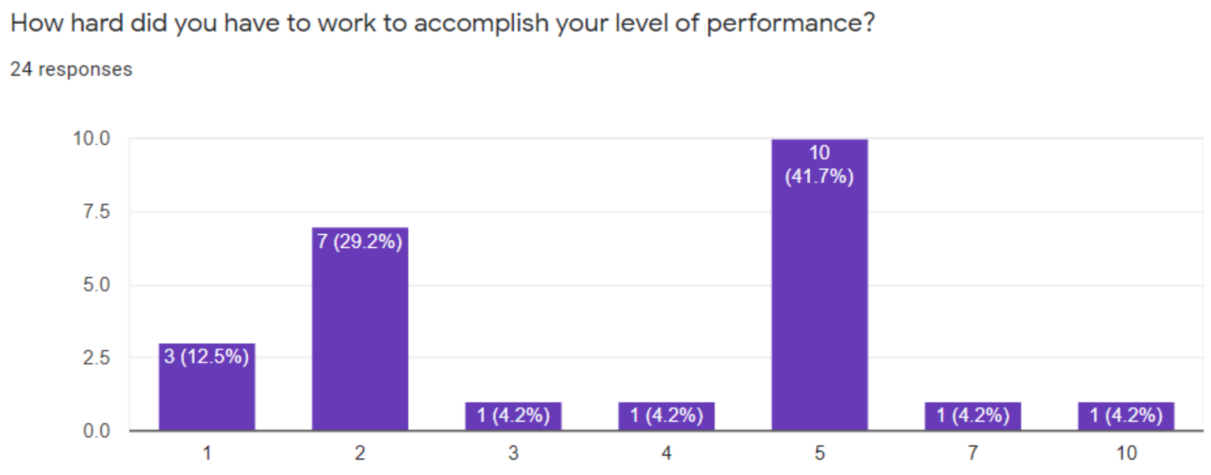


Figure A.21: Nasa TLX index question number 5

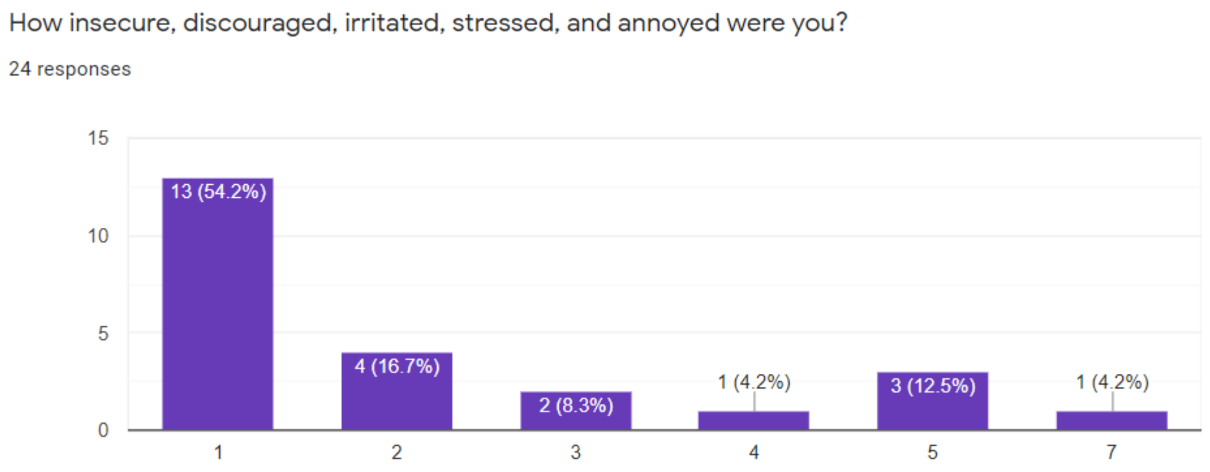


Figure A.22: Nasa TLX index question number 6