A Privacy-Preserving, Installation-Free Progressive Web App for Modern Slide Presentations

Damiano Perri(✉)

Department of Mathematics and Computer Science, University of Perugia, Perugia, Italy
damiano.perri@unipg.it

Abstract. This paper introduces an open-source Progressive Web App that empowers lecturers, researchers, and keynote speakers to deliver more modern and effective presentations, focusing on direct audience engagement and practical usability.

With privacy and simplicity at its core, the app requires no installation or setup, making its features instantly accessible and user-friendly.

The software is 220KB in size and is based entirely on web technologies that enable the management of PDFs, videos and images.

Thanks to the Window Management API, monitors connected to the computer are detected and indexed, and a second window of the same web page can be managed using the Broadcast Channel API, which is available on all modern browsers and provides local message exchange.

In this way, from a single codebase that opens the presenter's window, the programme can control a second browser window to display the presentation to the audience without transferring data over the internet, ensuring that no files from the presented documents are ever transmitted or stored on external servers.

Document rendering occurs entirely on the client side using the PDF.js library, released by Mozilla under an open-source license.

The built-in features include support for 'split' and 'bottom' PDF notes, real-time annotation and highlighting tools, and control of the presenter's webcam via local WebRTC.

If the user wishes, optional features can be enabled, such as remote control of the presentation via smartphone using WebSocket or an anonymous Questions & Answers module for the audience, these are the only features that require data exchange with the server after the initial loading of static files.

State-of-the-art analysis confirms there are currently no comparable web-based solutions. The application works on all major browsers and operating systems, making it a viable and accessible alternative to traditional tools.

Keywords: Learning Technologies · Web-based- Computing · Progressive Web App · Cloud Computing · Open Source · Human Computer Interaction

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2027
O. Gervasi et al. (Eds.): ICCSA 2026 Workshops, LNCS 16758, pp. 171–185, 2027.
https://doi.org/10.1007/978-3-032-30497-1_11

1 Introduction

This paper describes an innovative piece of software released under an open-source licence, designed to facilitate the management of school lessons, keynote speeches or presentations involving an audience to whom documents need to be shown. The project presented here has been developed using modern web technologies and does not require installation on the user's operating system. The technology underpinning the development of this application is based on HTML, CSS and JavaScript, and has enabled the creation of a Progressive Web App with a total size of 220KB. The features integrated into the application allow users to present PDF documents, videos and images, control the presentation remotely via a smartphone, navigate through all the slides of the PDF using a browser, manage Q&A sessions, manage timers for breaks, use a laser pointer or spotlight to highlight parts of the document, draw on the slides, use a dynamic zoom to enlarge parts of the document, and manage the presenter's webcam.

Throughout the development phase, we have consistently prioritised the need for a graphical interface that is both simple and effective one capable of displaying on the presenter's screen the information required to ensure their talk is delivered to the highest standard. Furthermore, native compatibility has been designed for PDFs generated by projects based on LaTeX Beamer, containing the speaker's notes on the right-hand side of the page, and for projects created using Google Slides, with the speaker's notes positioned at the bottom of the page and exported to PDF format using the platform's native functionality.

The main motivation behind this work lies in the limitations highlighted by current desktop applications and traditional PDF readers. Speakers and lecturers often find themselves working in diverse and restrictive environments, where they must install additional software or plugins to enable dual-screen support, such as the laptop screen and the projector screen used to deliver content to the audience.

By leveraging modern Web APIs, such as the Window Management API and the BroadcastChannel architecture, the Progressive Web App presented here uniquely bridges the gap between web and desktop applications. It alone guarantees native multi-monitor management from the browser and integrates smartphone remote control capability, all while maintaining minimal computational demands. Furthermore, the application embraces a typically 'local-first' philosophy: document rendering (handled by the PDF.js library) takes place entirely on the client side, and no file uploads to external servers are required, thus ensuring a smooth, immediate, secure experience that respects user privacy. This approach not only reduces latency, enabling real-time performance even without a stable internet connection, but also ensures that documents containing sensitive data or dealing with confidential matters can be used, as there is absolutely no transfer of data from the documents to a third-party cloud server under any circumstances. The introduction of interactive tools accessible via smartphone (remote control and Q&A module via WebSocket) further breaks down the physical barriers between the presenter and the audience, promoting

more dynamic and inclusive presentations without the need for dedicated or expensive hardware.

Finally, we chose to adopt an open-source approach to foster collaboration and continuous innovation, allowing developers from around the world to contribute to the project, improve existing features and adapt it to specific needs, thereby ensuring rapid evolution of the software in response to user requirements as well as enabling transparency and security of the source code.

This scientific article is organised as follows. Section 2 introduces the background and the motivations that led to the development of the software. Section 3 describes the software architecture by analysing the code and the client-server infrastructure that enables its operation. Section 4 analyses the results obtained and discusses possible future developments of the project.

2 Motivation and Related Works

This section reviews the current scientific literature and compares the software currently available for presenting documents to an audience, highlighting the limitations and challenges that led to the development of the software described in this article.

PDF is one of the most widely used formats for document exchange and for creating presentations, particularly in academic settings, thanks to its ability to preserve the formatting and appearance of the document across different platforms and devices. It was created in 1993 as a proprietary format developed by Adobe Systems [1], but was made public in 2008 and has since become an open standard ISO 32000-1:2008[1].

The current scientific literature is replete with papers analysing the technologies and software currently available for managing presentations and school lessons; generally, however, these works focus on installable software that serves as an alternative to PowerPoint, Keynote or Google Presentation, or concentrate on software for the interactive management of presentations, but generally with installable software dependent on the type of operating system used by the user and never with web applications.

In 2010, a piece of software called Pdf Presenter Console (PPC) [2][2] was released on GitHub, based on GTK, it provided support for multiple monitors simultaneously, but it has not been updated for years. This led to a fork, i.e. a derivative project, called pdfpc [3], which can be installed on Linux, macOS (via Homebrew) and Windows (via WSL), and is currently maintained and updated, but is not available as a web application. Installing PDFPC is not always possible, for example, you need to have administrator rights on the computer. The developers also offer the option to compile the software from source, but even in this case, you need to be reasonably familiar with the command line, and a whole series of libraries must be installed as a prerequisite, making the procedure inaccessible to the majority of users.

[1] https://www.iso.org/standard/51502.html.
[2] https://github.com/jakobwesthoff/Pdf-Presenter-Console.

Rico-Juan [4] presents an open-source framework for synchronising PDF presentations with video and audio, viewable in real time via the user's smartphone. Although this project is no longer maintained or updated, it represents an interesting and fundamental starting point for accessible systems. The software focused primarily on the user's experience and did not take into account aspects relating to the presenter, the management of their presentation, or the audience.

In 2014, Mozilla released a JavaScript library called pdf.js that allows PDF documents to be viewed directly within the browser, without the need to install plugins or additional software. To date, this library has reached version 5.0 and is widely used [5]. The advantage of this library is that it is open source, released under the Apache-2.0 licence via GitHub[3], and can be integrated royalty-free into commercial projects.

In 2015, Lenz et al. [6] suggested that Prezi software is a valid alternative to PowerPoint for creating presentations, particularly in an academic context, thanks to its ability to create more dynamic and engaging presentations, however, nothing is mentioned regarding privacy, security or the management of data contained within one's presentations. Today, Prezi also incorporates artificial intelligence features that can assist with studying, however, aside from a free version usable only for public presentations, it requires paid plans for personal use.

In 2016, Office Social [7] was launched, a plugin for PowerPoint that, via an app installable on a smartphone, allows the audience and the presenter to navigate through the slides of the presentation. The limitation of this software is its reliance on proprietary, closed-source software, the need to install plugins, and the requirement to install additional code on the user's smartphone. Furthermore, it does not provide any additional features that enhance the management of the presentation itself, but is limited to providing remote control and interaction with the audience. In the same year, a master's thesis was published in Austria [8] analysing the state of the art in presentation management software via interaction with mobile devices, highlighting the need for greater integration between devices, reduced complexity during installation, and a greater focus on the presenter's user experience. Avsar et al. [9] compare nine presentation management software packages, some free and others paid, highlighting the pros and cons of each. In the paper, they propose a new software package, called *Academic Presenter*, which is free and supports handwriting during the presentation. This software has not been updated since 1 October 2018 and the only supported operating system is Microsoft Windows.

In 2019, a paper [10] was published comparing Google Slides, Prezi, Tilda, Slides and Haiku Deck with PowerPoint. The comparison focuses on slide preparation and presentation delivery, and the authors note that most students tend to prefer Google Slides as a free and accessible alternative.

Scientific research has also analysed the impact in secondary schools, highlighting that presentations contribute to improving the quality of teaching; again

[3] https://github.com/mozilla/pdf.js.

in 2019, the main applications discussed were Microsoft PowerPoint, Keynote, Google Slides, Prezi, Tilda, Slides and Haiku Deck [11]. In recent years, modern web technologies have been increasingly used in the field of education, and many researchers have focused on how best to harness their full potential [12–14].

HTML is known as a markup language, which was subsequently complemented by CSS for graphics management and JavaScript for application logic management. The JavaScript programming language has evolved in three phases. The first is the introductory phase, which saw its inception and ran from 1995 to 2009, marked by the advent of AJAX technology and the ability to make web pages interactive. The second phase runs from 2009 to 2015, during which we witnessed the emergence of ES6 (ECMAScript 2015), numerous frameworks and libraries that simplified web application development—such as AngularJS and jQuery—as well as the introduction of classes, modules and asynchronous functions [15]. The third phase spans from 2015 to the present day, during which modern frameworks such as React, Vue.js and Next.js have made it possible to move application logic directly to the client, thereby also enabling the development of hybrid applications [16–18]. Furthermore, in recent years, the consolidation of cloud computing and the ability to deliver services to the public [19,20] via the network has led to the proliferation of web applications, shifting the computation of many everyday tasks to platforms external to local computing [21–24].

The literature review has highlighted that, to date, there is no open-source software based on web technologies that allows PDF presentations to be managed with advanced features such as those described in this article, and which is accessible from any device without the need to install additional software or plugins, whilst guaranteeing the privacy and security of the data contained in the presented documents.

3 Discussion

This section documents the technology underpinning the operation of the application presented, analysing both the code and the infrastructure that enables the service to be delivered to the public.

3.1 Design Decisions and Implementation Approach

Before commencing the software development phase, a requirements analysis and a feasibility study were carried out. The functionalities available to browsers are constantly evolving, and when building a web app, it is essential to consider what can be achieved and what limitations cannot be overcome.

There are five enabling technologies that have facilitated the development of this project. The first is the introduction of Web 2.0, which enabled the transition from static to dynamic pages thanks to AJAX. The second is the introduction of HTML5, which, through canvas, video and audio elements, allows complex objects to be rendered on a web page without requiring plugins, such

as the now obsolete Adobe Flash Player. The third is the JavaScript engine built into browsers, which allows code to be compiled into machine code that can be executed directly on the user's PC. This draws on the philosophy underlying Java, where a programme can run on any platform thanks to a virtual machine that interprets the code; in this case, it is the browser that includes the necessary functionality to interpret and execute the code. The fourth is the ability to use WebSockets, which enable real-time, two-way communication between client and server, thus allowing for the implementation of features such as remote control via smartphone and a Questions and Answers module between the audience and the presenter. Before WebSockets, communication could only take place via polling or long polling techniques, which involved periodically sending requests to the server to check for new data or keeping a communication channel open for a certain period of time while waiting for a message to arrive, however, these techniques were inefficient and did not guarantee real-time communication. WebSockets, on the other hand, allow a persistent connection to be established between client and server, enabling data to be sent and received in real time without the need to send periodic requests. Finally, the fifth and key technology is the ability to use the Window Management API to manage windows, the release of WebRTC for managing webcams across separate windows, and the ability to use the BroadcastChannel API to synchronise state across different windows, thereby enabling native multi-monitor management directly from the browser.

BroadcastChannel is a JavaScript API that enables communication between windows, tabs or iframes sharing the same origin (protocol, domain and port) and forms the basis of this project. Until a few years ago, managing multiple windows of the same web page was a highly complex task and practically impossible to achieve to the standards of quality and user experience this project aims to deliver. Users expect software that is simple, intuitive, immediate and allows them to get straight to work. The absence of any preliminary configuration or installation steps is, in fact, a key prerequisite for this project.

3.2 Backend Architecture

The server-side architecture has been designed to be extremely lightweight and limited to essential tasks, delegating all computational and graphical processing to the client device. The server is based on Node.js [25]. The backend is used exclusively to serve the static files of the Progressive Web App (HTML, CSS, JavaScript) on startup and, optionally, to manage slides remotely. For reasons of maintainability and security, the server was subsequently containerised using Docker, thus enabling rapid deployment based on images tagged with the release date. Docker is a modern containerisation platform that allows applications to be packaged with all their dependencies into a single container, ensuring consistency across different environments and simplifying deployment and maintenance [26–28].

The Node.js server is responsible for providing content that is predominantly static, with computations performed locally, however, it was chosen to enable

the exchange of messages with clients via WebSockets. It should first be noted that the use of WebSockets is entirely optional and serves only two specific functions, which are described below: remote control via smartphone and the Questions & Answers (Q&A) module between the audience and the speaker. For the presentation to function, once the static files have been downloaded, the application requires no further data from the server. Thanks to WebSockets, users can connect their smartphone to the presentation session and control the slides being shown in exactly the same way as they would using any remote control connected to the computer via USB. The connection is established by scanning a QR code containing a unique session token, which the user's device uses to connect to the server. The QR code consists of the server address and the GET parameter containing the session token. The QR code is generated using an open-source library for Node.JS[4].

A key aspect in terms of privacy and security is that the server processes only navigation commands and plain text; no PDF documents, images or sensitive data are ever uploaded, transmitted or stored on the server. Live sessions are protected and kept isolated through dynamically generated 64-character unique identifiers. Between the development phase and the delivery of content to the user, we also have a compilation phase based on Vite.js [29,30] to minimise JavaScript files, thereby reducing the overall size of the Progressive Web App to just 148KB. Adding the 72KB logo brings the total size to 220KB, ensuring fast loading even under sub-optimal network conditions.

3.3 Frontend Architecture

The frontend is the part of the application that is visible to the user and with which they can interact directly. It consists of three main parts: the Presenter Window, the Audience Window and the Remote Control Window. Once the PDF has been loaded into the system, the Presenter Window opens, this is the presenter's main window, displaying the slide navigator, the current slide, the presenter's notes (if any), and a preview of the next slide. An example of the window described above is shown in Fig. 1.

By utilising the *Window Management API*, the system is able to detect secondary monitors and automatically position the window in full-screen mode. Based on tests carried out, the automatic positioning feature is supported by Chrome, Edge and Opera. Firefox, Safari and lesser-known browsers that have not yet added support for this feature[5] have nevertheless been taken into account. During code development, a fallback was implemented for browsers with limited technical capabilities; the new window containing the audience view is positioned directly on the monitor used by the presenter, who can then move it to the desired screen. Firefox and Safari users will still be able to use all other features of the application correctly.

[4] https://github.com/soldair/node-qrcode.
[5] https://developer.mozilla.org/en-US/docs/Web/API/Window_Management_API.

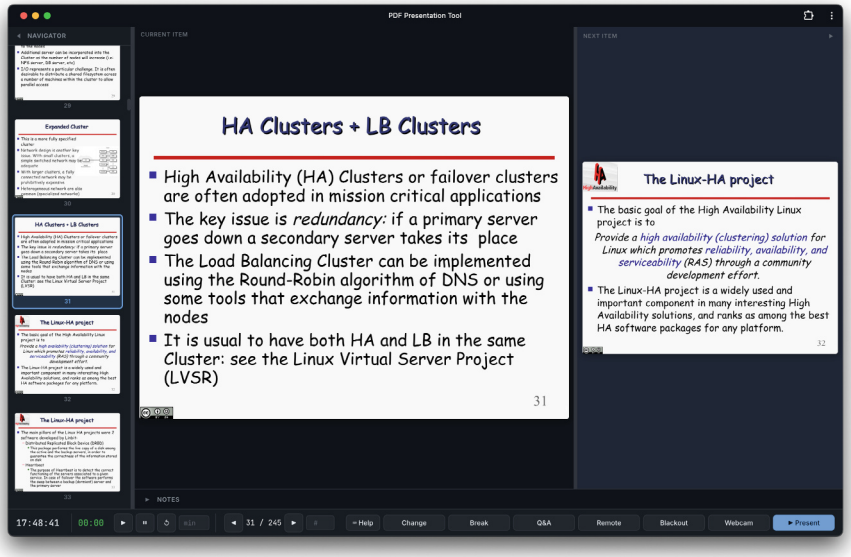


Fig. 1. A screenshot of the presenter's view, showing the navigation pane, the current slide, the next slide and the controls at the bottom of the window.

State synchronisation between the two displays occurs natively and instantly via the *BroadcastChannel API*, which has been natively supported by all major browsers since March 2022, as indicated in Mozilla's official documentation[6]. This API has enabled the implementation of the following features: navigation between slides; laser pointer mode, which helps indicate the presenter's mouse position on the audience's screen; zoom mode, which allows a section of the slide to be magnified by following the mouse cursor; spotlight mode, which highlights a section of the slide by darkening the background and simulating a torch illuminating parts of the slide; and whiteboard mode, which allows drawing on the slide and displaying the drawn lines in real time on the audience window. As communication is local, latency is extremely low, and slide changes are instantaneous. Client-side rendering of documents is handled by the *PDF.js* library.

A key feature of the application is the management of notes within PDFs. Currently, two types of layout are supported for the presenter's notes: 'split' and 'bottom'. The 'split' layout is typical of PDFs generated by LaTeX Beamer, where the left half of the page is dedicated to the slide and the right half to the presenter's notes. LaTeX Beamer is a LaTeX package [31] for creating presentations, which allows slides to be organised in a 'split' format with the speaker's notes on the right-hand side of the page and the slide on the left-hand side. The second supported layout is the 'bottom' layout, typical of PDFs generated by

[6] https://developer.mozilla.org/en-US/docs/Web/API/Broadcast_Channel_API.

Google Presentations, in which the slide is positioned at the top of the page and the presenter's notes are positioned at the bottom of the page, which is generally in A4 format. The slide is cropped automatically, and the notes are extracted directly from the text in the PDF and inserted into the presenter's window. To improve usability and the user experience for presenters using notes, buttons have been added that allow the text size of the notes to be enlarged or reduced, thereby improving readability.

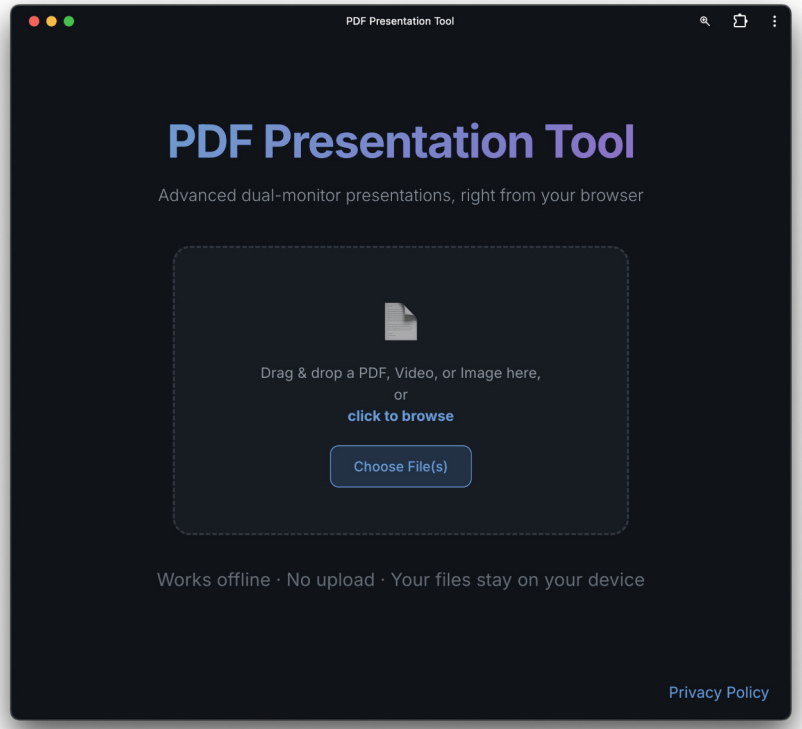


Fig. 2. The app's home screen, where you upload the file to be shared with the public. The screen is designed to be as simple and intuitive as possible.

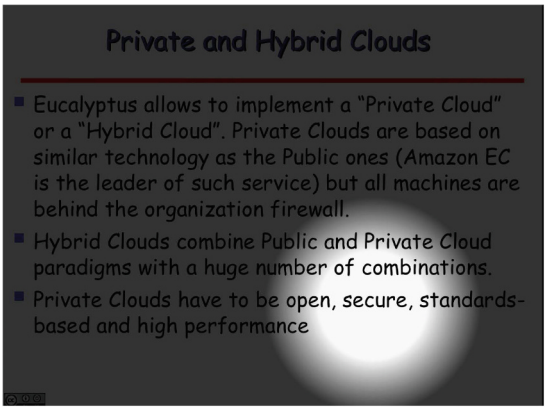
Finally, the presenter can activate their webcam and display an overlay bubble on the slide in one of the four corners. The audience will see both the slide and the presenter's face, thereby increasing engagement and connection with the presenter's talk. The webcam management feature does not require an external connection and therefore respects the presenter's privacy.

The camera's video stream is captured from the presenter's window and sent to the audience's window via the Session Description Protocol (SDP), again through the *BroadcastChannel* discussed previously.

3.4 Application Usage and Workflow

The user experience has been designed to follow a seamless workflow requiring no configuration. Starting a presentation is as simple as dragging and dropping the selected file (PDF or multimedia file) directly into the browser interface. Figure 2 shows the start screen, designed to be simple and intuitive.

Figure 3b shows the smartphone remote control screen, with a layout consisting of two large buttons that allow the user to move forward or backwards through the slides; the current slide number is displayed at the top. When the user changes slides, the device provides feedback by vibrating to indicate that it has received the command. To disconnect the remote from the presentation, simply close the browser tab; no further action is required.



(a) Figure A: Spotlight feature to highlight a section of the slide by darkening the background and simulating a torch illuminating parts of the slide.



(b) Figure B: Remote control screen via smartphone

Fig. 3. Overview of control and pointer functions.

Figure 3a, on the other hand, shows the spotlight feature, which allows you to highlight a portion of the slide by darkening the background and simulating a flashlight that illuminates parts of the slide. This feature is particularly useful for highlighting key concepts or directing the audience's attention to specific areas of the slide, thereby improving understanding and engagement during the presentation. Similarly, if audience interaction is requested, activating the Questions & Answers Session overlays a QR code on the projector screen that can be scanned by the audience.

After scanning it, the audience will be directed to a web page where they can enter text-based questions that will appear on the presenter's screen. Questions submitted by the audience are anonymous and are not stored or transmitted to third parties, thus ensuring user privacy. Closing the presentation automatically deletes all received questions, leaving no trace of sensitive or personal data.

Figure 4, on the other hand, shows the webcam management feature, which allows the presenter to activate their webcam and display an overlay bubble on the slide in one of the four corners. The same figure also shows the drawing feature; in fact, some marks have been drawn that the presenter is showing to the audience. As can be seen in the image, the presenter's face is centred within the bubble. This is achieved by cropping the image captured by the webcam to focus attention solely on what is relevant to the presentation, ignoring the background and other distracting elements.

The application can also be installed on the device. This is possible thanks to Progressive Web App technology, which allows a web application to be installed as if it were a native app, with a dedicated icon and the ability to be launched directly from the desktop or the device's app menu. This feature is compatible with Windows, macOS, and Linux as it is provided by the browser. The installation involved creating a manifest file and a service worker that manages the cache of static files subject to versioning. This means that if the user opens the application multiple times, the static files are loaded from the browser cache, reducing the load on the Node.js server and ensuring very fast loading times.

3.5 Performance Analysis

The software has been tested on various devices and operating systems. The test PDF consists of 245 pages and is part of the material provided in a university course; it contains both text and images and is 8.8 MB in size.

The first device tested is a MacBook Air with an Apple M1 processor, 8GB of RAM, and macOS Tahoe 15.2, connected to a projector via USB-C; this device is from January 2021. The first device took an average of 0.19 s to open the application and showed no slowdowns while using the test PDF, even with all features active simultaneously, including webcam management.

The second is an ASUS ZenBook 14 UM433DA-A5020T notebook running Linux Mint 21.2, with an AMD R5-3500U processor and 8GB of RAM, connected to a projector via HDMI; this device is from October 2020. The average application launch time was 1.1 s, and again, no slowdowns were observed while using the test PDF with all features active.

The application was then analysed using an independent test, LightHouse[7], an open-source third-party tool developed by Google to analyse a website's performance, accessibility, and other features. The test was performed on Chrome 146 on March 23, 2026, with a stable Wi-Fi connection, and the report returned scores of 99/100 for performance and 96/100 for platform accessibility.

[7] https://developers.google.com/web/tools/lighthouse.

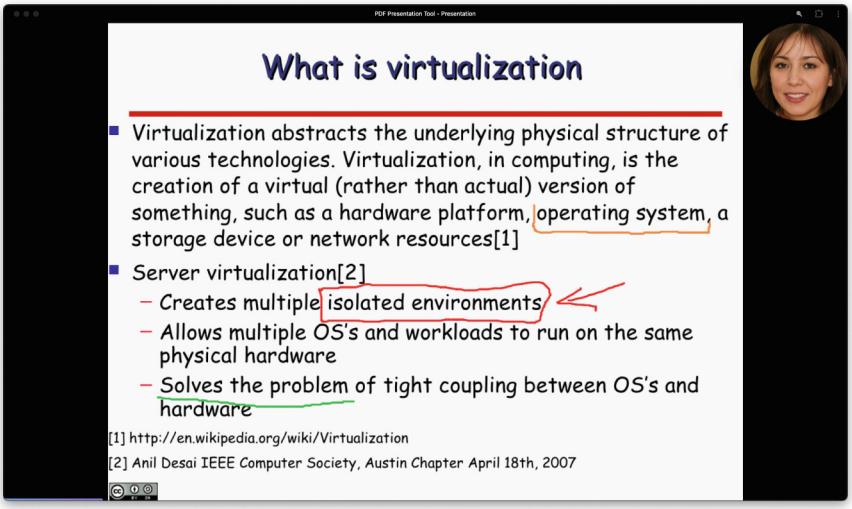


Fig. 4. The figure shows the window as seen by the audience. It illustrates the drawing and webcam management features, which allow the presenter to activate their webcam and display an overlay bubble on the slide in one of the four corners.

4 Conclusions and Future Work

This article presented an innovative open-source software called "PDF Presentation Tool"[8], designed to bridge the gap between traditional desktop applications and modern web technologies in educational and informational contexts. The application allows for the seamless and effective management of lessons, keynotes, or corporate presentations based on PDF files and multimedia content, making it accessible as it can be used on any device without the need to install additional software or plugins. All features are based on Web API and are therefore native to modern browsers.

By leveraging a "local-first" architecture based entirely on client-side processing, the system not only delivers high performance with low latency but also inherently ensures security and privacy by avoiding the upload of sensitive documents to third-party servers. With the exception of features requiring an internet connection (such as remote control via smartphone), this program can also be used offline; therefore, even in auditoriums with unstable connections, it will be possible to deliver presentations without interruptions or slowdowns caused by network latency. The project has successfully achieved its initial objectives, proving to be a resilient solution even in restrictive or heterogeneous educational and conference settings. It is hoped that in the coming years this tool will be integrated and used by an ever-increasing number of researchers, teachers, and professionals.

[8] https://presentations.damianoperri.it.

To ensure the transparency and security of the source code, as well as to encourage evaluation and contributions from the global scientific community, the entire project has been released in a public repository[9]. Moving forward, the project will focus on two main areas. First, we plan to conduct a survey to gauge user satisfaction and identify the features that faculty and students consider most urgent. To achieve this goal, we have already launched an open-access questionnaire designed to collect feedback based on the System Usability Scale and open-ended comments. The platform is currently in an open-access phase, during which anyone can access the site. So far, the platform has been used by 415 people, and initial feedback indicates widespread approval and a desire to continue using it. The testing phase will continue on a large scale for the next few months.

On the other hand, from a strictly technical perspective, we plan to expand the software by introducing native support for additional document exchange formats and implementing further features for smartphone-based remote control. Based on the tests conducted, the software is also compatible with Samsung DeX, the desktop environment that Samsung provides on its smartphones, however, as of now, two external monitors are not supported due to limitations in smartphone video outputs. It is hoped that in the future, with the evolution of smartphone video output technologies, it will be possible to extend compatibility to this type of device as well, thus making the software functional even without a computer, using only one's tablet or smartphone as a presentation device.

References

1. Cheng, W.-L.: Portable document format (pdf) — finally, a universal document exchange technology. J. Technol. Stud. **28.1** (2002)
2. Westhoff, J.: PPC: Pdf Presenter Console (2010). https://github.com/jakobwesthoff/Pdf-Presenter-Console
3. Westhoff, J., Vilar, D., et al.: PDFPC: a presenter console with multimonitor support for PDF files. Version 4.7.0 (2024). https://github.com/pdfpc/pdfpc
4. Rico-Juan, J.R.: Creating synchronised presentations for mobile devices using open source tools. In: Proceedings of the 2009 International Conference on e-Learning, E-Business, Enterprise Information Systems and eGovernment. Las Vegas, Nevada, USA: CSREA Press, pp. 50–52 (2009)
5. Qian, Y., Li, Y.: Web-based PDF Document Display Research and Implementation. Sci. J. Intell. Syst. Res. **7.7**, 28–37 (2025). https://doi.org/10.54691/r3n8m337. https://bcpublication.org/index.php/SJISR/article/view/8789
6. Lenz, P.H., McCallister, J.W., Luks, A.M., Le ,T.T., Fessler, H.E.: Practical Strategies for Effective Lectures. In: Annals of the American Thoracic Society 12.4, pp. 561–566 (2015). ISSN:2329-6933. https://doi.org/10.1513/AnnalsATS.201501-024AR. eprint: https://academic.oup.com/annalsats/article-pdf/12/4/561/66550575/annalsats12_4_561.pdf. https://doi.org/10.1513/AnnalsATS.201501-024AR

[9] https://github.com/DamianoP/PDF_Presentation_Tool.

7. Chattopadhyay, D., óHara, K., Rintel, S., Rädle, R.: Office Social: Presentation Interactivity for Nearby Devices. In: Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems. CHI '2016. San Jose, California, USA: Association for Computing Machinery, 2016, pp. 2487–2491. ISBN: 9781450333627. https://doi.org/10.1145/2858036.2858337
8. Schaffer, I.: Mobile Device Usage in Interactive, Co-located Presentations. English. MA thesis. 4232 Hagenberg, Austria: Interactive Media; FH Oberösterreich – Fakultät für informatik, Kommunikation und Medien (2016)
9. Avsar, B., Aliabadi, D.E., Aliabadi, E.E., Yousefnezhad, R.: Academic Presenter: a New Storytelling Presentation Software for Academic Purposes (2016). arXiv: 1607.06979. https://arxiv.org/abs/1607.06979
10. Osipovskaya, E.A., Burdovskaya, E.Y.: Presentation software tools in higher educational setting . In: ARPHA Proceedings 1, pp. 1487–1496 (2019)
11. Barabash, V., Osipovskaya, E., Dmitrieva, S., Budnik, E.: Cutting Edge Online Presentation Software In Higher Education. In: ICERI2019 Proceedings. 12th annual International Conference of Education, Research and Innovation. Seville, Spain: IATED, pp. 2812–2818 (2019). ISBN: 978-84-09-14755-7. https://doi.org/10.21125/iceri.2019.0720
12. Kerski, J.J.: Teaching and learning geography with a Web GIS approach. In: Revisioning Geography: Supporting the SDGs in the postCOVID era, pp. 113–135. Springer (2023)
13. Laganá, A., Gervasi, O., Tasso, S., Perri, D., Franciosa, F.: The ECTN virtual education community prosumer model for promoting and assessing chemical knowledge. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 10964 LNCS, pp. 533–548 (2018). https://doi.org/10.1007/978-3-319-95174-442
14. Wu, J.G., Zhang, D., Lee, S.M.: Into the brave new metaverse: envisaging future language teaching and learning. IEEE Trans. Learn. Technol. **17**, 44–53 (2023)
15. Gervasi, O., Perri, D., Simonetti, M.: Strategies and System Implementations for Secure Electronic Written Exams. In: IEEE Access 10, pp. 20559–20570 (2022). https://doi.org/10.1109/ACCESS.2022.3150860
16. Perri, D., et al.: Hybrid Methods for Automatic Collocation Extraction in Building a Learners' Dictionary of Italian. Computers **14.12** (2025). https://doi.org/10.3390/computers14120552
17. Perri, D.,et al.: Dockerized Architecture for a Progressive Web App: An Italian Collocations Dictionary. In: Lecture Notes in Computer Science 15891 LNCS, pp. 33–43 (2026). https://doi.org/10.1007/978-3-031-97617-93
18. Spina, S., Fioravanti, I., Zanda, F., Forti, L., Perri, D., Gervasi, O.: Developing a learner dictionary of collocations: description and evaluation of a multi-method approach. In: Corpus Linguistics and Linguistic Theory (2026). https://doi.org/10.1515/cllt-2025-0008
19. Fanfani, M., Palesi, L.A.I., Nesi, P.: DataDriven Mobility Demand-Versus-Offer Matching by Dynamic Integration of Public and Sharing Services. In: IEEE Access 14, pp. 23166–23193 (2026). https://doi.org/10.1109/ACCESS.2026.3661850
20. Perri, D., Simonetti, M., Gervasi, O.: Deploying Efficiently Modern Applications on Cloud. Electr. (Switzerland) **11.3** (2022). https://doi.org/10.3390/electronics11030450
21. Cone, L., Lai, S.S.: Infrastructural dependency in the datafied welfare state: the case of Google Chromebooks. Learn. Media Technol. **50.1**, 44–60 (2025). https://doi.org/10.1080/17439884.2024.2437680

22. Perri, D., Simonetti, M., Tasso, S., Ragni, F., Gervasi, O.: Implementing a Scalable and Elastic Computing Environment Based on Cloud Containers. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 12949 LNCS, pp. 676–689 (2021). https://doi.org/10.1007/978-3-030-86653-249
23. Bilotta, S., Palesi, L.A.I., Nesi, P.: Exploiting open data for CO2 estimation via artificial intelligence and eXplainable AI. Exp. Syst. Appl. **291**, 128598 (2025). ISSN: 0957-4174. https://doi.org/10.1016/j.eswa.2025.128598. https://www.sciencedirect.com/science/article/pii/S0957417425022171
24. Fereidooni, Z., Palesi, L.A.I., Nesi, P.: MultiAgent Optimizing Traffic Light Signals Using Deep Reinforcement Learning. In: IEEE Access 13, pp. 106974–106988 (2025). https://doi.org/10.1109/ACCESS.2025.3578518
25. Low, A., Siu, J., Ho, I., Liu, G.: Introduction to Node.JS. In: Proceedings of 24th Annual International Conference on Computer Science and Software Engineering. CASCON '2014. Markham, Ontario, Canada: IBM Corp, pp. 283–284 (2014)
26. Sobieraj, M., Kotyński, D.: Docker performance evaluation across operating systems. Appl. Sci. **14.15**, 6672 (2024)
27. Gervasi, O., Perri, D., Simonetti, M., Tasso, S.: Strategies for the Digitalization of Cultural Heritage. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 13382 LNCS (2022), pp. 486–502. https://doi.org/10.1007/978-3-031-10592-035
28. Singh, N., et al.: Load balancing and service discovery using Docker Swarm for microservice based big data applications. J. Cloud Comput. **12.1**, 4 (2023)
29. Hesti, E., Kaila, A.S., Handayani, A.S., Novianti, L., Rakhman, M.A.: An E-Commerce Platform for Coffee MSMEs: System Design and Basic Features. In: bit-Tech 8.1, pp. 910–920 (2025). https://doi.org/10.32877/bt.v8i1.2756
30. Asolo, E., Gil-Ozoudeh, I., Ejimuda, C., et al.: AI-Powered decision support systems for sustainable agriculture using AI-Chatbot solution. J. Digit. Food Energy Water Syst. **5.1** (2024)
31. Tantau, T., Wright, J., Miletic, V.: The BEAMER class User Guide (2016). https://doi.org/10.5555/3027684