Development and usability testing of a web application to streamline the creation of problems packages for BOCA-administered programming competitions

Luiz Gustavo Albuquerque dos Santos[1], Fabiola G. C. Ribeiro[1], Kenia S. de Oliveira[2]

[1]Department of Information Systems
Federal Institute Goiano – Catalão, GO – Brazil

[2]Federal Institute of Brasília – Brasília, DF – Brazil

gusalbukrk@outlook.com,
fabiola.ribeiro@ifgoiano.edu.br, keniasoli@gmail.com

Abstract. *BOCA is a programming contest management system widely used in Brazil. To register a programming problem in BOCA, a problem package consisting of a PDF problem statement and test case files for automated code assessment is required. However, the problem package creation feature in BOCA is rather limited. Therefore, in this paper, we present BOCA Problems Builder — a web application to streamline the creation of problems packages for BOCA-administered competitions. A key feature of the application is an extensive built-in catalog of programming problems sourced from the archives of the OBI and the Programming Marathon. A formative usability testing was conducted to evaluate the application and its results indicated a satisfactory user experience.*

1. Introduction

It is widely recognized in the academic literature that programming is a cognitively demanding activity, which presents substantial difficulties for both student learning and effective teaching [Xia 2017, Cheah 2020]. Computational Thinking (CT) is a essential skill for programming that encompasses the thought processes involved in applying skills such as algorithmic thinking, logic, abstraction, generalization, decomposition, and debugging to redefine complex problems into smaller, more manageable steps [Flórez et al. 2017]. Competitive programming is a pedagogical activity that can be used to foster the development of CT [Yuen et al. 2023, Audrito et al. 2023]. While the specific format of each competition differs depending on the organizer, at its core, competitive programming entails writing computer programs to solve a series of problems while competing with other programmers based on criteria such as program correctness, execution time, and development time [Majumdar 2017]. A programming competition problem consists of a name, a description, an explanation of the input data format and the expected output format, along with illustrative examples of both. In essence, it presents a computationally solvable problem designed to be tackled through an algorithm [da Cruz et al. 2022]. Research demonstrates a positive correlation between participation in programming competitions and desirable student outcomes, such as increased engagement and improved problem-solving and programming skills [Moreno and Pineda 2018, Piekarski et al. 2015, Yuen et al. 2023, Silva et al. 2023].

In Brazil, two prominent and long-established programming competitions are the Brazilian Olympiad in Informatics (OBI) and the Programming Marathon — both orga-

DOI: 10.5753/sbie.2024.244674

nized by the Brazilian Computing Society (SBC). The programming competition in the OBI has occurred annually since 1999 and currently is divided into four levels based on the students' grade level, ranging from junior, aimed at lower secondary school students, to senior, intended for first-year undergraduates. It consists of three subsequent phases — local, state-wide and national [Piekarski et al. 2023]. On the other hand, the Programming Marathon is a programming competition aimed at undergraduate and early graduate students that has been held since 1996. It consists of two phases: regionals and finals, with the latter including participants from across Latin America [Morais and Ribas 2019].

An efficient management of programming competitions necessitates a system capable of automating tasks such as problem statement distribution, submission assessment, and real-time ranking updates. The content management system that has been used in the Programming Marathon since the mid-2000s is the BOCA Online Contest Administrator (BOCA) [de Campos and Ferreira 2004, Morais and Ribas 2019], which was developed specifically to be used in the Programming Marathon but has since being used in non-official competitions [Algar Telecom 2012, Piekarski et al. 2015, Silva et al. 2023].

The creation of programming problems is a time-consuming and error-prone task [Sarsa et al. 2022, Zavala and Mendoza 2018]. Hence, programming contest organizers may opt to utilize existing problems from reputable sources rather than elaborating original problems. Nevertheless, the process of curating and formatting these existing problems to conform to the specifications of a specific contest management system remains time-consuming. In BOCA, it is necessary to provide a problem package when registering a problem for a contest. A problem package is a ZIP archive file containing — among other files — a Portable Document Format (PDF) file with the problem statement and text files with test cases for the automated assessment of code submissions. Among its many features, BOCA provides a rather limited problem package creator [de Campos 2024]. A thorough search for third-party alternatives failed to identify any graphical user application aimed at facilitating the creation of problems packages for BOCA-administered competitions, although a couple of command-line tools were found [Alves 2019, Nunes 2024]. However, it is important to note that generally only highly technical users prefer command-line tools over graphical applications due to the inherent usability issues of text-based interfaces [Vaithilingam and Guo 2019].

Therefore, in this paper, we present BOCA Problems Builder — an open-source web application to streamline the creation of problems packages for BOCA-administered programming competitions. The web application facilitate the generation of problems packages and its associated files such as the PDF file with the problem statement and the text files containing the test cases. Another key feature of the application is an extensive built-in catalog of programming problems, sourced from the archives of renowned programming competitions such as the OBI and the Programming Marathon. Furthermore, a formative usability testing with eight participants was conducted to evaluate the software in its initial iteration. The findings indicated that the application exhibited high usability and visual design quality — two crucial aspects of user experience (UX) — and was perceived as useful by the participants.

The remainder of this text is organized as follows. Section 2 outlines the key topics necessary for thorough comprehension of this paper. Section 3 provides a overview of the BOCA Problems Builder web application. Section 4 focuses on the experimental

methodology used to evaluate the developed software along with its results and discussion. Finally, Section 5 has the concluding remarks, next steps and future works.

2. Literature Review

The present section establishes a foundation in the two key topics necessary for a thorough understanding of this paper — the BOCA contest management system and UX.

2.1. BOCA

The main technical challenges of organizing a programming competition can be categorized into three parts: (1) problem creation, including all its related metadata such as statements, solutions and test cases; (2) contestant environment configuration, in particular with respect to environment consistency and network restrictions; and (3) contest management, i.e., problem statement distribution, automated grading with feedback, and real-time ranking updates [Maggiolo and Mascellani 2012].

Among the many programming contest management systems available, BOCA stands out as a very popular alternative in Brazil, due to its adoption in the Programming Marathon and open-source licensing [Morais and Ribas 2019]. For the registration of a problem in a competition, BOCA requires the upload of a problem package — a ZIP archive file containing a multitude of files needed for problem presentation, execution and assessment. The files related to code execution are standardized across problems packages, whereas the files pertaining to presentation and assessment are specific for each problem package. There are two problem presentation files: a text file named *problem.info* containing miscellaneous metadata and a PDF file containing the problem statement. In regards to the files related to problem assessment, these consist of pairs of input and output text files containing the test cases. During the automated code assessment, the submitted code solution will be executed as many times as there are input-output pair files. Each execution will utilize a specific input file, and the resultant output will be compared to the corresponding output file [de Campos 2024]. Often, a problem package will include several dozen input-output pair files to ensure thorough code assessment.

Programming competition organizers have the option to either elaborate original problems or utilize existing problems obtained from external sources. For programming competitions that necessitate the use of original problems, the creation of the problems packages and its files is entirely up to the organizers. On the other hand, organizers choosing to leverage existing problems from external sources utilize the files made available by others to assemble their problems packages. Both the OBI and the Programming Marathon offer public access to the problems from previous competitions on their respective websites [1]. More specifically, they provide access to the PDF files containing the problems statements for each past contest and the input and output files for each problem. A challenge encountered when curating programming problems from these sources is limited discoverability due to problems being categorized primarily by year which hinders efficient search and selection. Moreover, the PDF files obtained from the OBI and Programming Marathon include competition-specific branding. Consequently, organizers sourcing problems from these events may choose to create new PDF files with their own branding instead of reusing the original ones, thereby incurring an additional task.

[1] https://olimpiada.ic.unicamp.br/passadas/ and https://maratona.sbc.org.br/hist/index.html.

While primarily a content management system addressing the technical challenge three, BOCA also incorporates a problem package creation feature, which partially addresses the technical challenge one. However, such feature has a rather limited scope, as it does not handle common tasks involved in the creation of problems packages, such as the inclusion of multiple input-output pair files and PDF files generation.

2.2. UX

In the field of Human-Computer Interaction (HCI), usability and UX are two distinct but closely related concepts. As defined by [International Organization for Standardization (ISO) 2018], usability is the degree to which users can accomplish specific goals with effectiveness, efficiency, and satisfaction; whereas UX entails how users perceive and respond to a system in a specified context of use. It is clear that the two concepts overlap. However, UX is generally understood to be a broader concept that encompasses usability and all aspects of the interaction such as aesthetics, accessibility, and overall satisfaction [Lima et al. 2022].

In regards to evaluation techniques, usability testing is a common user testing method in which representative users perform a specified set of tasks to evaluate the usability of a system under the supervision of a facilitator. Usability tests can be categorized into two primary types: formative evaluation and summative evaluation. Formative evaluation involves collecting user feedback throughout development to guide iterative improvements, while summative evaluation is conducted at the end of development to gauge whether the final product meets the usability requirements. There is no consensus among researchers regarding the optimal number of participants for a usability test, but previous studies indicate that a participant pool of five to nine users is generally sufficient to identify approximately 80% of usability issues [Riihiaho 2018].

The think-aloud protocol and the use of questionnaires are two widely employed methods for data collections in usability testing [Riihiaho 2018]. Within the think-aloud protocol, participants are asked to verbalize their thought processes while performing assigned tasks. In regards to questionnaires, the use of standardized questionnaires is recommended because they offer greater reliability. The System Usability Scale (SUS) has emerged as the de facto standard questionnaire for usability evaluation. It comprises ten statements rated on a five-point Likert scale. The combined results are aggregated into a single number, ranging from 0 to 100, known as the SUS score; this score serves as a concise indicator of a system's overall usability. Recent studies report an average SUS score of around 70.8, with simpler tools often scoring higher, likely due to their inherent straightforwardness [Lewis 2018]. Another standardized tool commonly utilized in HCI is the Visual Aesthetics of Websites Inventory (VisAWI) — an 18-item questionnaire used to measure website aesthetics [Perrig et al. 2023]. In the original VisAWI, there are eight negatively-worded items. [Perrig et al. 2023] developed and validated a positive-item version named VisAWI-Pos to mitigate possible issues associated with negatively formulated items. Moreover, the Technology Acceptance Model (TAM) is an Information Systems theory used to understand the acceptance and use of a technology [Cheah et al. 2023]. According to the TAM, user acceptance of a technological innovation is predicated on two key factors: perceived usefulness (PU) and perceived ease of use (PEU). There are multiple versions of the the TAM model and related questionnaires. In [Davis 1989], the author proposed two distinct six-item questionnaires to separately assess PU and PEU.

3. BOCA Problems Builder

The BOCA Problems Builder was designated as a web application for the sake of user convenience since such applications are system-independent and do not require installation. A entirely client-side architecture was adopted to eliminate the need for a back-end infrastructure, thereby avoiding any hosting-related costs. Moreover, the interface and catalog of problems are to be initially offered exclusively in Portuguese.

3.1. Requirements

Software requirements can be categorized into functional and non-functional. Functional requirements (FR) define the specific capabilities the system must offer and non-functional requirements (NFR) describe the system properties and constraints (e.g., performance, security, usability, and maintainability) [Kurtanović and Maalej 2017]. The project initial iteration has a total of six requirements, which are listed below.

- **FR1**: problem management — i.e. the capability of view, create, edit and delete problems.
- **FR2**: creation of a catalog of problems sourced from the archives of the OBI and the Programming Marathon to enable users to quickly curate and import existing problems. The implementation of such feature involves the use of a web crawler. Web crawling is a technique for automated discovery and extraction of information from a designated group of web pages. Ethical web crawling must comply with a website's Robots Exclusion Protocol, which, through a *robots.txt* file, defines the permissions and limitations regarding the pages that web crawlers are allowed to mine for data [Gold and Latonero 2017]. During the requirements elicitation phase of this project, the *robots.txt* files from the OBI and Programming Marathon websites were examined, and no applicable restrictions on crawling were identified.
- **FR3**: contest metadata customization to allow users to edit parameters such as contest name and logo, which will be displayed in the generated PDF files.
- **FR4**: client-side persistence and backup features. Due to the application's purely client-side architecture, data must be stored within the web browser to ensure persistence across browser sessions. Moreover, the backup feature is crucial for enabling long-term data preservation and data transfer across multiple devices.
- **FR5**: PDF and ZIP files generation.
- **NFR1**: an intuitive and polished interface.

3.2. Implementation

The choice of the technology stack was primarily influenced by the popularity of the available alternatives. TypeScript [Microsoft 2024] was chosen as the programming language for the application. Moreover, a concise explanation of the implementation for each of the project's requirements is provided below.

- **FR1**. The problem management capabilities did not required third-party libraries.
- **FR2**. For web crawling, the command-line utility wget [Rühsen et al. 2024] was used to download all files related to previous competitions from the OBI and Programming Marathon servers. The remaining data extraction and transformation tasks were performed in Python with the following libraries: pdfplumber

[Singer-Vine 2024] for extracting data from PDF files, pypdf [Cimon et al. 2024] for splitting PDF files, and Googletrans [Han 2024] to translate the problems that were available exclusively in English. The scripts related to this requirement are versioned in the same repository as the web application source code. The resulting dataset is composed of over 600 problems extracted from the OBI (spanning the years 2009 to 2024) and the Programming Marathon (from 2013 to 2023). It is worth noting that the organization and accuracy of the extracted dataset is not yet optimal, and further work shall be done for further refinement.

- **FR3**. The contest metadata customization did not required third-party libraries.
- **FR4**. The implementation of persistence and backup functionalities relied on the IndexedDB database and the related third-party library Dexie.js [Lammes 2024].
- **FR5**. The libraries pdfmake [Pampuch and M. 2024] and JSZip [Knightley 2024] were utilized for the generation of PDF and ZIP files, respectively.
- **NFR1**. React [Meta Open Source 2024] and Bootstrap [Otto and Thornton 2024] were used for the development of the interface.

The application is available online [2] and a screencast is available on the project's source code repository to demonstrate the application in use.

4. Evaluation

This section focuses on presenting the experimental methodology, results and discussion of the usability testing used to evaluate the BOCA Problems Builder[3]. As it is usual for a formative usability test [Riihiaho 2018], the primary objective of the evaluation was to identify usability issues to enable incremental improvement. Moreover, a supplementary objective was to assess whether users perceive the application as fulfilling its intended purpose of streamlining the creation of problem packages.

Experimental design. During the testing phase, the participants were asked to complete 14 tasks pertaining to the following scenario "You are organizing a programming competition that will consist of three problems and you going to use the BOCA Problems Builder to generate the problems packages for this competition". The tasks were diverse and collectively encompassed every functionality available in the application at that point in time. The methods employed for data collection were the think-aloud protocol and a post-testing questionnaire. The participants' comments deemed pertinent were noted down by the facilitator and subsequently documented into a designated column within the spreadsheet containing the questionnaire responses. Upon completion of the the testing phase, a six-section questionnaire was administered to gather information about the participants and their feedback on BOCA Problems Builder. A brief description of the questionnaire sections is as follows: (1) impartiality and consent statements, requiring participants to confirm their ability to provide unbiased feedback and grant permission for data use in accordance with academic research ethics; (2) demographics and background experiences; (3) SUS questionnaire; (4) TAM questionnaire adapted from [Davis 1989]; (5) VisAWI-Pos questionnaire; and (6) grading of eight planned functionalities by order of importance and a open-ended field for participants to propose other functionalities and report any issues encountered. It is important to note that the SUS

[2] https://boca-problems-builder.netlify.app/.
[3] All files pertaining to the evaluation are publicly available.

questionnaire utilized was based on the translation and cross-cultural adaptation to Brazilian Portuguese proposed by [Lourenço et al. 2022]. In contrast, the TAM and VisAWI-Pos questionnaires were translated by the authors due to the lack of validated translations.

Participants. The usability test was administered to eight computer programming educators. Although the most representative users would have been teachers with prior experience in preparing problem packages for BOCA-administered competitions, no individuals with such background were available for the study. To mitigate this discrepancy between the participant and target user groups, a brief presentation was administered to each participant prior to their usability testing session. The presentation comprised introductory information on competitive programming, the BOCA contest management system, and the process of creating problem packages for BOCA-administered competitions.

Execution. The experiment was conducted in August 2024 at one of the Informatics Labs of the Federal Institute Goiano. The participants were tested individually, with each session lasting between 30 and 50 minutes — including the pre-testing presentation and the completion of the questionnaire.

Results. The evaluation participants comprised two males and six females, with a mean age of 38.5 years and a median age of 38 years. All participants held a degree in a Computer Science-related field and reported having taught or currently teaching programming courses to secondary and undergraduate students. Only three participants had experience organizing programming competitions.

The Figure 1 presents the scores derived from the SUS, TAM, and VisAWI-Pos questionnaire sections. The TAM and VisAWI-Pos scores were normalized to a 0-100 scale with min-max normalization to facilitate data presentation. Of particular interest are the mean scores of 90.6, 91.1, 88.0 and 82.4.

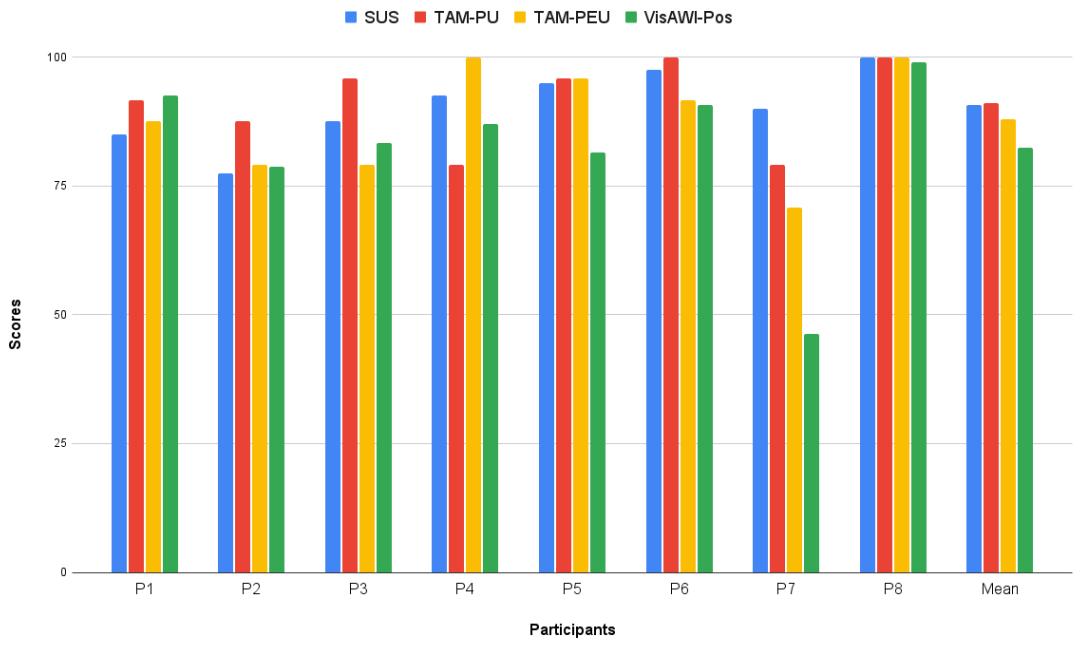

Figure 1. The SUS, TAM-PU, TAM-PEU, and VisAWI-Pos scores

In the sixth and concluding section of the questionnaire, participants were requested to rate the importance of the planned functionalities using a 5-point scale. The results indicated that the features were prioritized in the following order: mechanism to filter problems by difficulty level (mean score of 5), customization of the instructions printed to the booklet (4.75), client-side routing for improved navigation (4.25), English language support (4.125), integration of a text editor with formatting capabilities (4.125), addition of programming problems from other sources (4.125), general visual design improvements (3.875), and dark mode support (3.75). Furthermore, all but one participant highlighted the absence of visual indication following form submission. This feedback emerged as one of the most significant observations gathered in the testing, due to both its prevalence and the strong emphasis participants placed on the need for such a feature.

Discussion. The findings were overwhelmingly positive for the BOCA Problems Builder. The SUS, TAM-PEU, and VisAWI-Pos scores indicate that the application exhibits a high level of usability and a polished visual design. Specifically regarding the SUS score, is of particular note that the obtained mean falls within the A+ grade on the Sauro-Lewis curved grading scale [Lewis 2018]. As discussed in the literature review, usability and aesthetics are critical factors in UX. Hence, the strong performance in these areas suggests that the application provided a positive UX. Furthermore, the TAM-PU score demonstrates that participants perceive the application as useful. Regarding the planned features ranking findings, it is noteworthy that all functionalities received relatively high ratings, which substantiate their pertinence. Lastly, the absence of a visual indicator following user actions on the interface was a significant oversight, given the ubiquity of this feature in modern applications, and it should be addressed as a priority.

Threat to validity. The absence of individuals with prior experience preparing problems packages for the BOCA system among the participants impacted the study's ability to accurately represent the target user group. Nevertheless, participants underwent a brief pre-testing educational intervention to mitigate this limitation.

5. Conclusions, Next Steps and Future Works

This paper presented the development and usability testing of the BOCA Problems Builder. The results from the evaluation indicated that the application provides a satisfactory UX. Moreover, a secondary but significant contribution of this research is the public release of a dataset comprising programming problem data extracted from the OBI and Programming Marathon archives, previously available only in unstructured format.

It is important to note that this paper is a work in progress. The next step is to address identified issues and incorporate additional functionalities in response to the feedback obtained from the formative testing. Subsequently, a summative testing with a larger and more representative participant pool will be conducted to evaluate the application in its final iteration and to conclusively validate the results from the formative evaluation.

In regards to future works, the public availability of the programming problems dataset compiled in the present research can be used as a foundation for subsequent endeavors. Potential avenues for future exploration include: in-depth data analysis to identify trends and patterns in problem difficulty and topic distribution, integration of the dataset into coding learning platforms, and implementation of machine learning techniques for automated problem generation or problem solving.

Acknowledgements

This paper is a result of the research of the scientific initiation conducted with the support of the Institutional Program for Scientific and Technological Initiation Scholarships (PIBIC) subsided by the Federal Institute Goiano.

The authors acknowledge the use of generative AI tools, specifically ChatGPT 4o and Gemini 1.5 Flash, as revision aids in the writing of this paper.

Data Availability Statement

The BOCA Problems Builder source code is available here. The programming problems dataset created during this research is available here. The files related to the evaluation are available in Portuguese here.

References

Algar Telecom (2012). Roteiro para configuração de minimaratona de programação. https://docplayer.com.br/36306096-Minimaratonas-de-programacao.html.

Alves, E. (2019). ejtools. https://gitlab.com/ejudge/ejtools.

Audrito, G., Ciobanu, M., Laura, L., et al. (2023). Giochi di fibonacci: Competitive programming for young students. *Olympiads in Informatics*, pages 19–31.

Cheah, C. S. (2020). Factors contributing to the difficulties in teaching and learning of computer programming: A literature review. *Contemporary Educational Technology*, 12(2):ep272.

Cheah, W. H., Jusoh, N. M., Aung, M. M. T., Ab Ghani, A., and Rebuan, H. M. A. (2023). Mobile technology in medicine: development and validation of an adapted system usability scale (sus) questionnaire and modified technology acceptance model (tam) to evaluate user experience and acceptability of a mobile application in mri safety screening. *Indian Journal of Radiology and Imaging*, 33(1):36–45.

Cimon, L., Thoma, M., and Peveler, M. (2024). pypdf. https://github.com/py-pdf/pypdf.

da Cruz, A. K. B. S., Neto, C. d. S. S., da Cruz, P. T. M. B., and Teixeira, M. A. M. (2022). Utilizaçao da plataforma beecrowd de maratona de programaçao como estratégia para o ensino de algoritmos. In *Anais Estendidos do XXI Simpósio Brasileiro de Jogos e Entretenimento Digital*, pages 754–764. SBC.

Davis, F. D. (1989). Perceived usefulness, perceived ease of use, and user acceptance of information technology. *MIS quarterly*, pages 319–340.

de Campos, C. P. (2024). BOCA: BOCA Online Contest Administrator. https://github.com/cassiopc/boca.

de Campos, C. P. and Ferreira, C. E. (2004). BOCA: um sistema de apoio a competições de programação. In *Workshop de Educação em Computação*. Sociedade Brasileira de Computação.

Flórez, F. B., Casallas, R., Hernández, M., Reyes, A., Restrepo, S., and Danies, G. (2017). Changing a generation's way of thinking: Teaching computational thinking through programming. *Review of Educational Research*, 87(4):834–860.

Gold, Z. and Latonero, M. (2017). Robots welcome: Ethical and legal considerations for web crawling and scraping. *Wash. JL Tech. & Arts*, 13:275.

Han, S. (2024). Googletrans. https://github.com/ssut/py-googletrans.

International Organization for Standardization (ISO) (2018). Ergonomics of human-system interaction—part 11: Usability: Definitions and concepts (iso 9241-11: 2018).

Knightley, S. (2024). JSZip. https://github.com/Stuk/jszip.

Kurtanović, Z. and Maalej, W. (2017). Automatically classifying functional and non-functional requirements using supervised machine learning. In *2017 IEEE 25th International Requirements Engineering Conference (RE)*, pages 490–495. Ieee.

Lammes, S. (2024). Dexie.js. https://github.com/dexie/Dexie.js.

Lewis, J. R. (2018). The system usability scale: Past, present, and future. *International Journal of Human–Computer Interaction*, 34(7):577–590.

Lima, D. T., Moura, F. R. T., Alves, A. V. N., de Moura Parracho, T., Zacarias, R. O., dos Santos, R. P., and da Rocha Seruffo, M. C. (2022). Ux-tracking: Web and multimodal tool for user experience evaluation. In *Anais Estendidos do XXVIII Simpósio Brasileiro de Sistemas Multimídia e Web*, pages 107–110. SBC.

Lourenço, D. F., Carmona, E. V., and Lopes, M. H. B. d. M. (2022). Tradução e adaptação transcultural da system usability scale para o português do brasil. *Aquichan*, 22(2).

Maggiolo, S. and Mascellani, G. (2012). Introducing CMS: a contest management system. *Olympiads in Informatics*, 6:86–99.

Majumdar, A. (2017). MacACM: Encouraging competitive programming via mentorship and outreach. *XRDS: Crossroads, The ACM Magazine for Students*, 24(1):14–15.

Meta Open Source (2024). React. https://react.dev/.

Microsoft (2024). TypeScript. https://www.typescriptlang.org/.

Morais, W. B. and Ribas, B. C. (2019). Maratona-Linux: um ambiente para a Maratona de Programação. *Anais do Computer on the Beach*, pages 416–426.

Moreno, J. and Pineda, A. F. (2018). Competitive programming and gamification as strategy to engage students in computer science courses. *Revista ESPACIOS*, 39(35).

Nunes, D. (2024). ds-contest-tools. https://github.com/danielsaad/ds-contest-tools.

Otto, M. and Thornton, J. (2024). Bootstrap. https://github.com/twbs.

Pampuch, B. and M., L. (2024). pdfmake. https://github.com/bpampuch/pdfmake.

Perrig, S. A., von Felten, N., Honda, M., Opwis, K., and Brühlmann, F. (2023). Development and validation of a positive-item version of the visual aesthetics of websites inventory: The visawi-pos. *Indian Journal of Radiology and Imaging*, pages 1–25.

Piekarski, A. E., Miazaki, M., Hild, T., Mulati, M. H., and Kikuti, D. (2015). A metodologia das maratonas de programação em um projeto de extensão: um relato de experiência. In *Anais dos Workshops do Congresso Brasileiro de Informática na Educação*, volume 4, page 1246.

Piekarski, A. E. T., Miazaki, M., da Rocha Junior, A. L., Militão, E. P., and da Silva, J. V. P. (2023). Programação competitiva em um projeto de extensão para o ensino técnico em informática. *Revista Conexão UEPG*, 19(1):1–14.

Riihiaho, S. (2018). Usability testing. *The Wiley handbook of human computer interaction*, 1:255–275.

Rühsen, T., Shah, D., and Scrivano, G. (2024). GNU Wget2. https://www.gnu.org/software/wget/.

Sarsa, S., Denny, P., Hellas, A., and Leinonen, J. (2022). Automatic generation of programming exercises and code explanations using large language models. In *Proceedings of the 2022 ACM Conference on International Computing Education Research-Volume 1*, pages 27–43.

Silva, T. R. d. M. B., Braga, G., Silva, M. A. L., Araújo, M., et al. (2023). Maratonando! inspirando e capacitando programadores com diversidade de gênero e variedade de competições. In *Anais do XVII Women in Information Technology*, pages 346–351. SBC.

Singer-Vine, J. (2024). pdfplumber. https://github.com/jsvine/pdfplumber.

Vaithilingam, P. and Guo, P. J. (2019). Bespoke: Interactively synthesizing custom guis from command-line applications by demonstration. In *Proceedings of the 32nd annual ACM symposium on user interface software and technology*, pages 563–576.

Xia, B. S. (2017). A pedagogical review of programming education research: what have we learned. *International Journal of Online Pedagogy and Course Design (IJOPCD)*, 7(1):33–42.

Yuen, K. K., Liu, D. Y., and Leong, H. V. (2023). Competitive programming in computational thinking and problem solving education. *Computer Applications in Engineering Education*, 31(4):850–866.

Zavala, L. and Mendoza, B. (2018). On the use of semantic-based aig to automatically generate programming exercises. In *Proceedings of the 49th ACM technical symposium on computer science education*, pages 14–19.