


Automated Assessment of Simple Web Applications

Luís Maia Costa ✉ 

Faculty of Engineering, University of Porto, Portugal
CRACS – INESC TEC, Porto, Portugal

José Paulo Leal ✉ 

Faculty of Sciences, Universidade do Porto, Portugal
CRACS – INESC TEC, Porto, Portugal

Ricardo Queirós ✉ 

uniMAD, ESMAD/P.Porto, Portugal
CRACS – INESC TEC, Porto, Portugal

Abstract

Web programming education is an important component of modern computer science curricula. Assessing students' web programming skills can be time-consuming and challenging for educators. This paper introduces Webpal, an automated assessment tool for web programming exercises in entry-level courses. Webpal evaluates web applications coded in HTML, CSS, and Javascript, and provides feedback to students. This tool integrates with Virtual Learning Environments (VLEs) through an API, allowing the creation, storage, and access to exercises while assessing student attempts based on the created exercises. The evaluation process comprises various subcomponents: static assessment, interface matching, functional testing, and feedback management. This approach aims to provide feedback that helps students overcome their challenges in web programming assignments. This paper also presents a demo showcasing the tool's features and functionality in a simulated VLE environment.

2012 ACM Subject Classification Applied computing → Interactive learning environments

Keywords and phrases Web Applications, Static Assessment, Interface Matching, Functional Assessment, Feedback Manager

Digital Object Identifier 10.4230/OASICS.ICPEEC.2023.11

Category Short Paper

Funding José Paulo Leal is financed by National Funds through the Portuguese funding agency, FCT – within project LA/P/0063/2020. We also would like to acknowledge the European Union's Erasmus Plus programme (agreement no. 72020-1-ES01-KA226-VET-096004).

1 Introduction

The internet is a crucial resource for learning and researching nowadays, enabling teachers and students to access and share information. In the 1990s, Virtual Learning Environments (VLEs) were introduced to aid education, making it easier to share knowledge and provide feedback to students [10]. Evaluating students' programming abilities can be challenging since most exercises have an uncountable number of solutions. Moreover, since students must complete many programming tasks, it can burden teachers to assess them all [2]. This challenge increases in the automated assessment of web applications since these often use a combination of programming and markup languages, such as HTML, CSS, and JavaScript. Additionally, events in web applications do not follow a predetermined sequence and may occur randomly. As a result, conventional black-box methodologies that rely on test cases to assess student performance are inadequate for analyzing web interfaces and graphical user interfaces



© Luís Maia Costa, José Paulo Leal, and Ricardo Queirós;
licensed under Creative Commons License CC-BY 4.0

4th International Computer Programming Education Conference (ICPEEC 2023).

Editors: Ricardo Alexandre Peixoto de Queirós and Mário Paulo Teixeira Pinto; Article No. 11; pp. 11:1–11:8

OpenAccess Series in Informatics



OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

11:2 Automated Assessment of Simple Web Apps

(GUIs) [2, 12]. To address these issues, this paper introduces a tool under development called Webpal (WEB Programming Assessment for Learning) that allows teachers to automatically assess basic web programming skills involving HTML, CSS, and JavaScript. While there are many tools and frameworks available to automate the assessment of various programming languages, there is a need for more solutions for web page evaluation. This work aims to develop an npm package that integrates with VLEs and assists in the automated assessment of web programming assignments and is capable of providing incremental feedback to the students about their attempts. The paper is structured as follows. Chapter 2 provides an overview of the current state of the art on tools and algorithms comparable to Webpal. Chapter 3 describes the Webpal architecture, features, and evaluation process. Lastly, Chapter 4 summarizes the significant contributions and highlights potential future directions for research in this area.

2 State of the Art

In this study, we introduce a tool for evaluating basic web applications, specifically involving HTML, CSS, and JavaScript. To the best of our knowledge, no equivalent tools exist that can offer an automated assessment within these distinct programming languages. As such, our analysis focuses on tools that embody attributes relevant to our project, particularly those relating to interface and functional assessments. It is noteworthy that most of the prevailing literature in the interface assessment domain seems to come from the early 2000s, with a predominant emphasis on tools developed in Java.

Douce, Livingstone, and Orwell [3] reviewed various automated assessment tools and pointed out some advantages and disadvantages. On the positive side, since assessing programming assignments is laborious, these tools can reduce teachers' work, allowing them to concentrate on more critical tasks, such as clarifying CS and programming concepts. Also, computers are less likely to fail if configured correctly, while humans are error-prone. On the other hand, there are many restrictions on what can be assessed automatically, and the feedback provided by these tools needs to be revised for educational reasons [11].

In 2010, Ihantola et al. [6] presented a literature review of the current systems for automatically assessing programming assignments from 2006 to 2010. The authors recommended that new automated assignment systems should explain clearly how the tool functions and that the security of these systems should get more consideration. They also suggested that automated assessment tools should be open-source to avoid scattering, emphasized the need to integrate automated assessment into virtual learning environments, and identified the assessment of web applications as an area for future research.

It's possible to identify two types of assessment for programming assignments:

- Dynamic Assessment assesses a program by executing it. It requires a secure sandbox environment that doesn't interfere with the host's computer integrity. It's possible to check the program functionality by comparing the result with test data sets or its efficiency by monitoring execution-related activities such as execution time and memory usage [2].
- Static Assessment, contrarily to dynamic, assesses programming code without executing it. Adequate software code doesn't just guarantee proper execution; other critical elements must be considered, such as coding style and code smells, to check if the code contains the correct syntax. It can also check for plagiarism in the source code [2, 1, 9, 16].

2.1 Interface Assessment

In 2006, Gray and Higgins [5] introduced an approach to grading GUIs by analyzing the hierarchical relationship among interface components. This method extracts data from graphical elements and uses dynamic class loading in object-oriented programming languages to retrieve information about the interface's object structure. Then, a file specifies the required actions and objects to test and grade assignments.

Štěpánek and Šimková [17] described three algorithms for comparing the internal structure of HTML trees to detect duplicate interfaces (phishing) and assist in education. The first algorithm counts the number of occurrences of each element in a tree. The second algorithm calculates a ratio representing the percentage of reference paths in the compared interface. The third algorithm identifies the largest possible subtree in both interfaces and calculates a ratio between them.

Thackston [15] noted in 2020 that only some tools are available to grade web assignments. The author proposed a method that uses XPath queries to automate the assessment of HTML files. Although XPath has the potential for automatic evaluation, the author reported limitations in flexibility, incomplete coverage, and grading difficulties when the assignment is partially correct.

Leal and Primo [12] proposed a web interface matching algorithm that uses element mapping to identify and compare elements from different interfaces. This method extracts original and derived properties from the Document Object Model (DOM) API, reflecting spatial relationships between elements. The authors suggested refining the algorithm and extending it to evaluate interfaces visually and functionally, along with creating an incremental feedback manager.

2.2 Functional Assessment

Sztipanovits et al. [14] proposed an application to assess the functionality of web graphical user interfaces (GUIs). This solution requires a list of URL hosting submissions with the interfaces to evaluate and an Excel document defining the inputs and test cases.

The js-assess online playground [7] is a tool that automatically assesses Javascript programming exercises. It is a serverless tool that runs inside a Javascript engine and combines an online editor with libraries to evaluate functionality, style, programming errors, and software metrics. js-assess can evaluate Javascript functionality, style, programming errors, and software metrics, being unable to assess interfaces and styling. Due to the lack of a server, feedback cannot be stored, making it useful only for self-studying. This tool has a collection of assignments available and is implemented in pure HTML and Javascript, facilitating the integration of js-assess into any web page. The authors highlighted the importance of researching how Javascript should be taught, the libraries that should be used, and where Javascript should be introduced to students.

Mirshokraie [8] proposed an automated technique to generate test cases for individual Javascript functions and DOM event sequences, which focused on developing efficient and practical tests for JS web applications. The approach consists of a three-step process that involves dynamically exploring the application to deduce a test model, generating test cases, and creating test assertions automatically using mutation testing. The technique uses specific mutation operators to identify common JS programming errors and analyze only the application's necessary code. The study results showed that the generated tests covered an average of only 68.4%, demonstrating that this method is not particularly interesting in generating tests for web applications.

11:4 Automated Assessment of Simple Web Apps

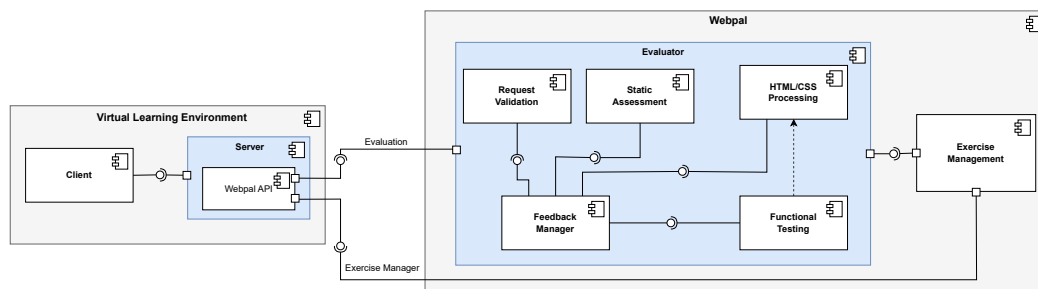
WebWolf [13] is a framework for automatically evaluating introductory web programming exercises. This tool can load web pages, find and inspect elements, click links, and perform unit testing. Teachers can create JUnit-like Java programs to test websites by writing test methods to simulate user inputs on the page and make assertions to compare the student's result with the expected output. However, WebWolf requires knowledge of Selenium's WebElement and the specification of IDs on web elements, which is not ideal. The authors concluded that this framework significantly reduces grading time compared to manual grading and has the potential for automated assessment.

2.3 Feedback Manager

There is a lack of information on systems incorporating incremental feedback managers. ProtoAPOGEE [4] is an automated grading system that aims to guide students and enhance faculty productivity by providing informative and iterative feedback through failed unit testing. The tool comprises three modules: project specification, automatic grading, and grading report viewer. The system requires teachers to know the Ruby programming language to write scripts. One challenge of providing detailed feedback to students was the need for teaching staff to write lengthy descriptive messages about errors. ProtoAPOGEE addresses this issue by generating an animation demo for each failed test case, providing a step-by-step guide on how the student failed a particular requirement. Teachers can also provide textual hints to assist students during the assignment. The tool can provide feedback at two levels: a summary report and detailed feedback for each requirement and test case.

3 Webpal

Webpal automates the evaluation process for web programming exercises in beginner-level courses. It assesses web applications coded in HTML, CSS, and Javascript, and provides immediate non-repetitive feedback to students. It consists of an independent module responsible for managing the entire tool, from file parsing to assessment and feedback generation. Webpal interacts with VLEs via an API, enabling the creation, storage, and access to exercises and assessing student attempts based on created exercises. The evaluation process encapsulates several subcomponents, including the analysis of the submission structure, code syntax verification, generation of a paired array of HTML elements from both the solution and attempted interfaces, and element mapping for functional testing. The latter employs chai/mocha tests provided during the exercise creation to evaluate the attempted web application. Each process can yield messages for the feedback generation manager, who processes these messages to provide the most helpful feedback for assisting students in overcoming their challenges.



■ Figure 1 Webpal architecture.

3.1 Webpal API

The Webpal API provides a collection of functions for managing and evaluating web development exercises. These functions include:

- `createExercise(exerciseData, testData, assignment)`: Returns a unique identifier after creating an exercise using the provided exercise data, test data, and assignment information.
- `getExerciseData(id)`, `getExerciseTestData(id)`, `getExerciseAssignment(id)`: Each function fetches specific information about an exercise using its unique identifier.
- `getAllExercises()`: Returns a list of all existing exercises.
- `deleteExercise(id)`: Deletes an exercise using its unique identifier.
- `evaluateAttempt(exerciseID, attemptData, PORT, previousFeedback)`: Evaluates an exercise attempt and returns the feedback.

The exercise and attempt data follow a structured JSON format where each submission consists of an array of objects. Each object includes two main properties: the filename and code. The filename is a string that represents the name of the file being submitted, while the code contains the actual content of the file, such as HTML, CSS, or JavaScript code. The test data, represented by a string of Mocha tests written in JavaScript, is used to evaluate the functionality of a learner's submission. Presently, Webpal assesses one HTML file per submission, including linked CSS and JavaScript files. Our future goal is to enable the evaluation of multiple pages per student submission. Webpal is deployed as an npm package, which simplifies its integration in existing JavaScript-based systems, and requires a server to execute.

3.2 Static Assessment

A preliminary step of Webpal `evaluateAttempt` function involves static evaluation. Three npm packages¹ are used for validating code syntax for each file type. For HTML, `html-validator` is employed, which validates HTML using Nu HTML Checker or `html-validate`. For CSS, `w3c-css-validator` is used, taking advantage of the W3C public CSS validator. Finally, `JSHINT` identifies Javascript errors and suggests better coding practices in JS. Static assessment helps to pinpoint issues before executing the evaluator, providing feedback about these problems without requiring Webpal to run the evaluator entirely, enriching students' understanding of their solution, and teaching them ways to produce more interoperable code.

3.3 Browser Emulation

Evaluating interfaces requires web page emulation to determine the relationships and positions of elements. Alternative approaches, such as `jsdom` and `cheerio`, were considered, which attempt to implement and emulate web standards on the server side. While these tools manage and verify relationships between HTML elements, they cannot simulate element positions. Webpal employs `Playwright`, a web testing and automation framework that runs headless browser instances, providing accurate results through the use of an actual web page. Another similar framework is `Puppeteer`, but it's older and lacks certain features, such as cross-browser support. The `express` npm package is used to serve files, offering a simple

¹ <https://www.npmjs.com/>

11:6 Automated Assessment of Simple Web Apps

solution for routing individual web pages and websites. Webpal examines the extensions of the specified files and express serves the HTML files locally on a port selected by the VLE. If CSS and JS files are imported into an HTML page, express will serve them. This dynamic routing approach enables the creation of various evaluators using different ports simultaneously.

3.4 Evaluation

The HTML/CSS processing component is responsible for processing attempt and solution HTML files, creating two tree structures representing HTML elements and associated CSS styles. This is achieved through a recursive traversal of elements and their children, ensuring all nested components are processed and included. The comparison and evaluation processes are simplified by utilizing a list of elements. Comparisons between elements and their properties are based on the Leal and Primo web interface matching algorithm [3]. This algorithm examines spatial relationships, styling, and textual content, resulting in a computed match score for each element pair within the attempt and solution trees. Pairs with the highest match scores are then selected. Subsequently, the functional testing component generates a mapping based on the best pairs, connecting the original IDs of the solution elements to their corresponding attempt element IDs. This mapping updates the functional test data provided initially (JS file with chai/mocha written tests), facilitating the automated testing of the attempt web page.

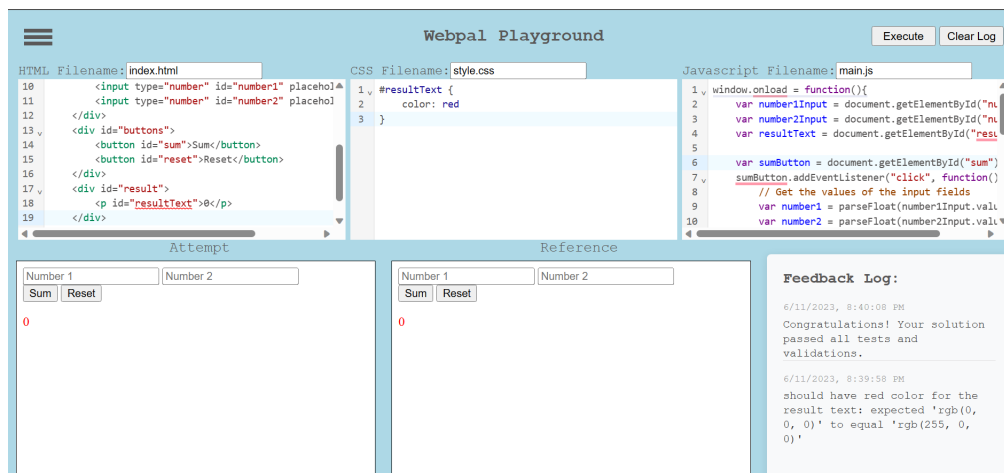
3.5 Feedback Manager

This module comprises a class that handles outputs sent by the request validation, static assessment, and functional testing modules. The Webpal execution follows a specific order, so it doesn't need complete execution if errors are encountered in the first steps of its pipeline. First, the student's attempt is checked against the predefined JSON schema to ensure it follows the required format. If the solution doesn't corroborate with the necessary structure, Webpal generates a feedback message highlighting this problem. Next, the static assessment module performs a syntax validation on the student's code. If any syntax errors are detected, a feedback message related to that error is generated. After the static evaluation and the HTML tree generation, the functional testing module tries to map the HTML elements of the student's attempt to the solution's HTML elements. The student is advised to review their HTML structure and element relationships if Webpal cannot find any correspondence between the two sets of elements. Finally, the functional testing module evaluates the functionality of the student's code, generating feedback for any failed tests.

3.6 Webpal Playground

The Webpal Playground is a demo created in Vue.js to showcase the features and functionality of Webpal, emulating a simplified Virtual Learning Environment. The demonstration consists of two pages: Playground and Backoffice. It is complemented by a backend server running on Node.js using the Webpal API. The Playground is the primary interface for students to interact with and explore Webpal. On this page, students can select from a list of available exercises and work on them using a built-in code editor. The code editor is divided into three sections for HTML, CSS, and Javascript, allowing students to edit and preview their work in real-time. It's also possible to check the solution output on this page. Students can execute and submit their code to be evaluated by clicking the "Execute" button. When

the code is submitted, Webpal evaluates the attempt and provides feedback to the student. The feedback is displayed in the “Feedback Log”, which records all feedback messages along with their timestamps. The Backoffice is designed for exercise management. Teachers can create new exercises on this page by providing an assignment name, exercise file names and code, and chai/mocha tests to evaluate the student’s code. Once an exercise is submitted, a success message is displayed, and the exercise is added to the list of available exercises in the Playground. The backend server utilizes the Webpal API to handle various tasks, such as creating, deleting, and fetching exercises, as well as evaluating student attempts. This server communicates with the frontend pages using HTTP requests and responses. Figure 2 illustrates the demonstration interface, presenting a scenario where two feedback messages are generated in response to two distinct attempts made by a student.



■ **Figure 2** Webpal playground demonstration.

4 Conclusion and Future Work

In this paper, we presented Webpal, an automated assessment tool for web programming exercises in introductory-level courses. Webpal focuses on evaluating student submissions coded in HTML, CSS, and Javascript, providing incremental feedback. The tool integrates with VLEs through an API. This work’s main contribution is an approach to assess the main aspects of simple web applications through static evaluation, interface matching, and functional testing. This approach also ensures that students receive valuable feedback to improve their programming skills and address the challenges they face in their assignments. Another contribution of this work was the development of an npm package that simplifies integration with VLEs and will serve as a base for validating this approach. There are several potential future directions for research in this field. It would be beneficial to extend the capabilities of Webpal to cover more advanced web programming assignments, such as multi-page web applications, or develop plugins to integrate directly with platforms like Moodle or Blackboard. The next steps include validating Webpal by integrating it into a code playground to evaluate the effectiveness of Webpal in terms of student learning outcomes and satisfaction to get insights of its impact. Comparing the performance of students using Webpal against traditional assessment methods could help establish the benefits of automated web programming assessment.

References

- 1 Kirsti Ala-Mutka, Toni Uimonen, and Hannu-Matti Järvinen. Supporting students in C++ programming courses with automatic program style assessment. *JITE*, 3:245–262, January 2004. doi:10.28945/300.
- 2 Kirsti M Ala-Mutka. A survey of automated assessment approaches for programming assignments. *Computer Science Education*, 15(2):83–102, 2005. doi:10.1080/08993400500150747.
- 3 Christopher Douce, David Livingstone, and James Orwell. Automatic test-based assessment of programming: A review. *J Educ Resour Comput*, 5, September 2005. doi:10.1145/1163405.1163409.
- 4 Xiang Fu, Boris Peltsverger, Kai Qian, Lixin Tao, and Jigang Liu. Apogee-automated project grading and instant feedback system for web based computing. *SIGCSE'08 – Proceedings of the 39th ACM Technical Symposium on Computer Science Education*, pages 77–81, 2008. doi:10.1145/1352135.1352163.
- 5 Geoffrey R. Gray and Colin A. Higgins. An introspective approach to marking graphical user interfaces. *ITiCSE06 – Proceedings of the 11th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education*, 2006:43–47, 2006. doi:10.1145/1140123.1140139.
- 6 Petri Ihantola, Tuukka Ahoniemi, Ville Karavirta, and Otto Seppälä. Review of recent systems for automatic assessment of programming assignments. In *Proceedings of the 10th Koli Calling International Conference on Computing Education Research*, Koli Calling '10, pages 86–93, New York, NY, USA, 2010. Association for Computing Machinery. doi:10.1145/1930464.1930480.
- 7 Ville Karavirta and Petri Ihantola. Serverless automatic assessment of javascript exercises. *ITiCSE'10 – Proceedings of the 2010 ACM SIGCSE Annual Conference on Innovation and Technology in Computer Science Education*, page 303, 2010. doi:10.1145/1822090.1822179.
- 8 Shabnam Mirshokraie. Effective test generation and adequacy assessment for javascript-based web applications. *2014 International Symposium on Software Testing and Analysis, ISSTA 2014 – Proceedings*, pages 453–456, 2014. doi:10.1145/2610384.2631832.
- 9 Jan Nordin. A review on the static analysis approach in the automated programming assessment systems. *National Conference on Programming 07*, 2007. URL: <https://www.researchgate.net/publication/228328534>.
- 10 RL O'Leary and AJ Ramsden. *Virtual Learning Environments*, pages 1–30. Economics Learning and Teaching Support Network, 2002.
- 11 José Carlos Paiva, José Paulo Leal, and Álvaro Figueira. Automated assessment in computer science education: A state-of-the-art review. *ACM Transactions on Computing Education*, 22:1–40, September 2022. doi:10.1145/3513140.
- 12 Marco Primo and José Paulo Leal. Matching user interfaces to assess simple web applications. *OpenAccess Series in Informatics*, 91:7:1–7:0, 2021. doi:10.4230/OASIcs.ICPEC.2021.7.
- 13 Antonio C. Siochi and William R. Hardy. Webwolf: Towards a simple framework for automated assessment of webpage assignments in an introductory web programming class. *SIGCSE 2015 – Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, pages 84–89, 2015. doi:10.1145/2676723.2677217.
- 14 Mate' Sztipanovits, Kai Qian, and Xiang Fu. The automated web application testing (awat) system. *Proceedings of the 46th Annual Southeast Regional Conference on XX, ACM-SE 46*, pages 88–93, 2008. doi:10.1145/1593105.1593128.
- 15 Russell Thackston. Exploring the use of xpath queries for automated assessment of student web development projects. *SIGITE 2020 – Proceedings of the 21st Annual Conference on Information Technology Education*, pages 255–259, 2020. doi:10.1145/3368308.3415389.
- 16 Nghi Truong, Paul Roe, and Peter Bancroft. Static analysis of students' java programs. *IFAC Symposium on Advances in Control Education*, pages 317–325, January 2004.
- 17 Jiří Štěpánek and Monika Šimková. Comparing web pages in terms of inner structure. *Procedia – Social and Behavioral Sciences*, 83:458–462, 2013. doi:10.1016/j.sbspro.2013.06.090.