

A novel and light-weight learning platform for digital workbooks (not only in computer science)

André Greubel ¹

Abstract: In this demo, we present a novel, browser-based, interactive workbook system designed for learning situations in which students should be able to start quickly, work independently, and preserve their progress without the usual administrative overhead of a full learning-management system. The central goal is to make modern learning resources easily available to students and teacher alike. In the platform, three interactive workbooks for computer science education are currently available to try. The source is open (available under <https://github.com/greubela/edusquirrel-workbook>), a hosted version (or information on where a new version moved to) is available at <https://evadid.it/workbooks>.

Keywords: Learning platform, Computer Science Education, Learning Management System

1 Motivation and Goals

Digital learning environments can “effectively [improve] student performance, satisfaction, and engagement” [GAB22] and provide a wide range of features like dashboards, learning analytics, forums, content repositories, assignment workflows, and integrations with institutional systems. While these features provide value, they also introduce friction [Ni94]: Teachers must create courses, manage participants, configure permissions, structure content, publish resources, and sometimes train learners before the actual learning activity begins. In Practice, typical “heavy-duty” content management platforms like Moodle [Mo] that are frequently used [Bi25] require extensive background infrastructure and creating content on these platforms (or sharing content with multiple courses) is not straight-forward.

In many educational situations like workshops, project days, or open online material, the main need is different: students should be able to open a link and start learning. Resources like Inf-Schule [Fr16], CS Unplugged Materials [Be09; LSR19], or Leifiphysik [FW25] provide low friction entry points for learning. However, they typically do not provide full learning sequences and are frequently very limited in the richness of their functionality.

The central goal of our platform is to be a middle-ground between those extremes: a platform that makes accessing digital content easy, while also offering more interactive functionality and longer learning sequences. The focus hereby is primarily on the access to content: while it should be possible to adapt existing or create new content, different from resource platforms like Lehrer Online [Ed], the main goal is not to exchange as much content as possible. Instead, content should be curated and only become publicly available online after a certain level of quality was verified by the creators of the platform.

¹ Humboldt-Universität zu Berlin , Institut für Informatik , Lehrstuhl Informatik und Gesellschaft ,
andre.greubel@hu-berlin.de,  <https://orcid.org/0000-0002-7915-6520>

2 Design of the Platform

There are two central goals for our platform. First, material should be easy to use (“low floor”, see [Re05]). Starting with the material should be as easy as opening an URL in the browser – no popups, no registration required. Also, the platform should not rely on an extensive backend that requires maintenance or can shut down.

Because of the this requirement, the platform is almost completely client-based: The whole application essentially is one large directory with HTML, Javascript, and additional resource files. Hosting the platform is as easy as copying the directory to a web server and configuring the base path of the index.html (relative to the server root) in a config file. An internet connection and additional backend server can *extend* the core functionality (e.g., by forwarding requests to an LLM after clicking on a help button), but are never required for key functionality (it is technically possible to serve the workbooks using a local web server).

Second, it should use modern features that other, simpler platforms lack. For example, it is possible 1.) to easily create material for different languages, 2.) to use LLM feedback on free-text exercises (that do not spoil the solution), 3.) the current work progress is automatically stored in the cache of the browser (to prevent loss of data by a browser reload or crash), 4. and it is also possible to download it to a json file or send it (together with a name and an re-access code) to an SQL database. Note that this is optional behavior that requires both an internet connection and a suitable database set up by the teacher.

To create workbooks, theree offers several content templates (called `WorkbookElement`]). The `Workbook` itself is created via a JSON file containing a nested structure of these elements.

On the one side, this includes static content like structuring elements (like workbook sections or groups of exercises) and content elements (like text formatted as html or images based on a source URL), including several defaults for typical layouts (like headlines or safety warnings). These elements can be based on an `LanguageMapContentId`: these ids get resolved with the JSON files in the resource directory `LanguageMaps` before rendering and can be used to provide content in multiple languages (see Figure 1 as example).

On the other side, this includes basic interaction elements (called `WorkbookInteraction`) like single and multiple choice elements, input areas for text (both inline and as text area) and numbers (both integers, fractionals, and algebraic expressions (currently implemented as `AlgebriteNumber` [Ca21]), as well as exercises for reordering elements or sorting them into given categories. The content of these interactions can be language sensitive, too, and changes to their state are automatically logged and synchronized to the local browser cache.

Aside from these general-purpose exercise templates, there are also more complex interaction elements focused on computer science education. Two we want to highlight here are the `GptInteraction` and `TurtleStitchRecreateShapeInteraction` (shown in Figure 2).

```
/myWorkbook/map-en.json /myWorkbook/map-de.json /myWorkbook/map-universal.json
{
  "Sec1Ex1":
    "Do something **nice**",
    "Sec1Img1Url":
      "/images/sec1ImgEn.png",
    "Sec1ImgDesc":
      "English Captions"
}
{
  "Sec1Ex1":
    "Mach etwas **schönes**",
    "Sec1Img1Url":
      "/images/sec1ImgDe.jpg",
    "Sec1ImgDesc":
      "Deutscher Untertitel"
}
{
  "Sec1Img2Url":
    "/images/sec1Img.png",
    "Sec1Ex1":
      "Do something **nice**",
    "Sec1ImgDesc":
      "[exercise not available in your language yet]"
}
```

Fig. 1: Potential files in the /LanguageMap/ directory. In the JSON defining the workbook, it would be possible to create elements with ImageElement(ID(myWorkbook/Sec1Img1Url)) and MarkdownText(ID(myWorkbook/Sec1Ex1)). If a key does not exist in the currently active language (e.g., when selecting Danish), the value associated with the key in map-universal is used as fallback.

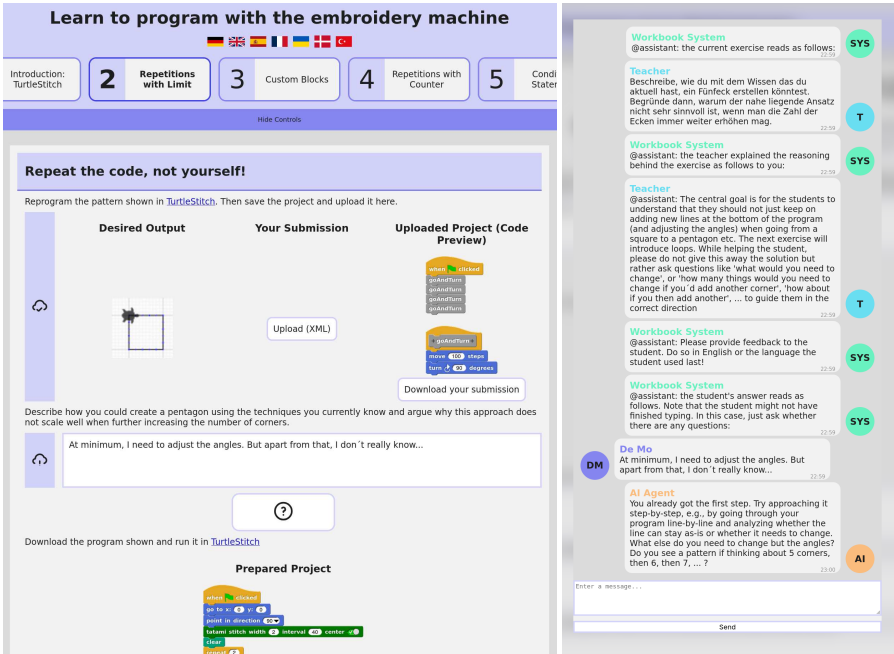


Fig. 2: Two cropped screenshots from the EmbroideryWorkbook. [Left] The main view of a workbook showing the beginning of the second section, including two interactions (a TurtleStitchRecreateShapeInteraction, and a TextInteractionWithHlmscaffolding). Note that the first interactions is fully and the other only partially synced to the local storage and default remote database. The preview of the TurtleStitch program is not an image, but dynamically rendered and thus always displayed in the currently selected language. [Right] When clicking on the help button, a Messenger appears to offer LLM feedback (if available). This messenger is automatically prompted with available information, including hints from the teacher (provided as LanguageMapContentId). In practice, messages starting with “@assistant” or “@teacher” are not visible to the student. The hosted demo version of the workbook has a backend connection enabling a very small number of tokens per IP-address. A custom connection requires a reverse proxy server and an OpenAI API-key.

3 Implementation Details

The project is implemented in roughly 18.000 lines of Scala 3 code using laminar [Bu] for the reactive interface. It is open source and cross-compiled to JavaScript via Scala.js [Do13].

The architecture of the platform including the overall structuring of the data, central data and control flows, and the transaction-based interaction format were engineered by the authors. Code assistant tools (like Codex [Fi22; Op26]) were at several points used to write specific implementations of transactions or elements. However, most of the resulting lines needed to be modified or re-factored before they could be merged into the main branch (esp. to prevent duplication of functionality or inconsistent architecture choices). The initial hope of using such tools to adapt existing open source platforms like finite state automaton simulations to be usable from within our platform was discarded as unreasonable.

The architecture of the project itself is divided into four modules: core (most of everything), client (ui-specific code), server (backend specific code), and worker (a small facade that enables running more complex computations within a web worker). Communication between the frontend and the web worker backend, as well as the optional server backend are implemented using `ExecutionCommandFactory` that can create an `ExecutionCommand` to be transmitted to an `ExecutionClient` (e.g., the backend server for transmitting LLM requests or exporting session data to a database). All Interaction with the workbook are transaction-based using the `State[Type]` classes (or its laminar `Var[Type]` equivalent).

4 Future Work

Content currently publicly available from within includes three workbooks focusing on computer science: learning to program with TurtleStitch, understanding data compression for everyday use, and building a self-watering plant with an Arduino. More content is in development. The focus of future developments has the following main directions:

- We want to provide a workbook editor so that more teachers (not only from computer science) are able to create material for our platform.
- Provide a lightweight dashboard for analysis for a teacher in which they can visualize the state of all students that submitted their state to the configured database and e.g., verify whether one of the LLM feedbacks contained an error.
- We plan to introduce more types of content are planned (e.g., by providing an integrated IDE and unit tests for programming exercises, or by allowing for for exercises based on functional modeling or sequenced games like SQL Island [Sc])
- Improvements to the aesthetics of the page and a nicer and consistent color palette.

Overall, our digital platform provides a novel, open-source platform for learning content with ready-to-use material for computer science education. The demo itself will be focused on letting visitors play around with the hosted version to get a more detailed impression.

References

- [Be09] Bell, T. et al.: Computer Science Unplugged: School Students Doing Real Computing Without Computers. *The New Zealand Journal of Applied Computing and Information Technology* 13 (1), pp. 20–29, 2009.
- [Bi25] Bitkom e. V.: 8 von 10 Lehrkräften haben Zugang zu digitalen Lernplattformen, <https://www.bitkom.org/Presse/Presseinformation/8-von-10-Lehrkraeften-Zugang-digitalen-Lernplattformen>, Published: 21.01.2025. Accessed: 11.07.2026, 2025.
- [Bu] Bulatov, N.: Laminar: Simple, Expressive, and Safe Reactive UI Library for Scala.js, <https://laminar.dev/>, Accessed: 11.07.2026.
- [Ca21] Casa, D. D.: Algebrite: Computer Algebra System in JavaScript, <https://github.com/davidedc/Algebrite>, Version 1.4.0. Accessed: 11.07.2026, 2021.
- [Do13] Doeraene, S.: Scala.js: Type-Directed Interoperability with Dynamically Typed Languages, tech. rep. EPFL-REPORT-190834, Accessed: 01.03.2026, École Polytechnique Fédérale de Lausanne (EPFL), 2013.
- [Ed] Eduversum GmbH: Lehrer-Online, <https://www.lehrer-online.de/>, Accessed: 11.06.2026.
- [Fi22] Finnie-Ansley, J. et al.: The robots are coming: Exploring the implications of openai codex on introductory programming. In: *Proceedings of the 24th Australasian computing education conference*. Pp. 10–19, 2022.
- [Fr16] Froitzheim, M. et al.: Nutzung eines elektronischen Schulbuchs im Informatikunterricht am Beispiel des Projektes inf-schule. In: *DeLFI 2016–Die 14. E-Learning Fachtagung Informatik*. Gesellschaft für Informatik eV, pp. 71–82, 2016.
- [FW25] FWU Institut für Film und Bild in Wissenschaft und Unterricht gGmbH: Über LEIFIphysik, <https://www.leifiphysik.de/ueber-leifiphysik>, Accessed: 11.07.2026, 2025.
- [GAB22] Gamage, S. H. P. W.; Ayres, J. R.; Behrend, M. B.: A systematic review on trends in using Moodle for teaching and learning. en, *Int. J. STEM Educ.* 9 (1), p. 9, 2022.
- [LSR19] Lindner, A.; Seegerer, S.; Romeike, R.: Unplugged Activities in the Context of AI. In: *International conference on informatics in schools: Situation, evolution, and perspectives*. Springer, pp. 123–135, 2019.
- [Mo] Moodle Pty Ltd: Moodle: The World’s Open Source Learning Platform, <https://moodle.org/>, Accessed: 11.06.2026.
- [Ni94] Nielsen, J.: *Usability engineering*. Morgan Kaufmann, 1994.
- [Op26] OpenAI: Codex, AI coding agent. Version accessed on 2026-06-11, 2026, <https://github.com/openai/codex>.
- [Re05] Resnick, M. et al.: *Design Principles for Tools to Support Creative Thinking*, tech. rep., Workshop report, revised October 30, 2005, National Science Foundation Workshop on Creativity Support Tools, 2005.
- [Sc] Schildgen, J.: SQL Island, <https://sql-island.informatik.uni-kl.de/>, Accessed: 11.07.2026.