Leopold-Franzens University Innsbruck
Faculty of Mathematics, Computer Science and Physics

Department of Computer Science

Quality Engineering

Bachelor's Thesis

in partial fulfillment of the requirements for the degree of

Bachelor of Science

Design and Implementation of a Web-Based, Block-Based Learning Platform with AI-Assisted Grading for Children Aged 8–12

by

Fabio Plunser
(Student ID: 0123456789)

Submission Date: October 8, 2025
Supervisor: Ass. Prof. Dr. Michael Vierhause

Declaration of Authorship

I hereby declare under oath that I have completed this bachelor's thesis independently and without unauthorized assistance. I have not used any sources or aids other than those indicated. All passages that have been taken from publications or other sources, either verbatim or in paraphrase, have been marked as such. I agree to the archiving of this bachelor's thesis.

Place and Date: _____

Signature: _____

Contents

List of Figures

Abstract

This thesis presents the design and prototypical implementation of a web-based learning platform that teaches introductory programming concepts to children aged 8–12 using a focused, block-based visual environment. In contrast to open-ended ecosystems such as Scratch or MakeCode, the platform offers tightly scoped, auto-graded exercises with clear learning goals per task. Two exercise types are supported in the minimum viable product (MVP): input/output (I/O) tasks and turtle graphics. Each exercise defines a constrained toolbox, optional starter code, multi-step hints, a reference solution, and a deterministic test specification. A client-side sandbox executes learner code safely and an AI-inspired (rule-based and structured) grading layer provides immediate feedback without uploading user code to external services. The work contributes (i) a requirements model for age-appropriate block tasks, (ii) a modular architecture combining Blockly, a secure sandbox, and extensible graders, (iii) an authoring JSON schema enabling reproducible content, and (iv) an evaluation of feasibility through exemplar exercises. Pedagogical depth (empirical learning outcome measurement) is explicitly out of scope; the focus is on technical design, safety, extensibility, and accessibility.

Acknowledgments

I thank my supervisor Ass. Prof. Dr. Michael Vierhause for guidance.

1 Introduction

1.1 Motivation

Digital literacy and computational thinking have become core competencies in lower secondary education. Existing block-based environments (e.g. Scratch) are powerful but often too open-ended for time-constrained classroom deployment, while commercial offerings can be closed, hardware-bound, or lack granular auto-grading. Educators need a lightweight, privacy-friendly platform delivering small, focused programming challenges with immediate formative feedback.

1.2 Problem Statement

There is a gap between open creative playgrounds and rigid quiz systems: teachers lack a flexible, self-hostable solution that (a) constrains complexity per exercise, (b) supports deterministic grading without server execution of arbitrary code, and (c) remains accessible to learners aged 8–12.

1.3 Objectives

The objective is to design and prototype a modular platform that:

- Supports authoring and execution of structured block-based programming exercises.
- Provides instant, deterministic auto-grading for I/O and turtle tasks.
- Ensures safety via sandboxing and minimal data retention (privacy by default).
- Enables localization (German/English) and age-appropriate UI/UX.
- Is extensible for future exercise types and custom blocks.

1.4 Research Questions

1. How can a constrained block-based exercise model balance expressiveness and cognitive load for ages 8–12?
2. What architectural approach enables deterministic, safe, client-side execution and grading?
3. How should an extensible JSON schema for exercises encapsulate toolbox, tests, hints, and localization?

4. What are the technical trade-offs of client-only sandboxing versus server-side execution?

1.5 Scope

In scope are: two exercise types (I/O, turtle), at least ten curated tasks (variables, loops, conditionals, simple functions), a JSON-based authoring format, bilingual content, sandboxed execution, and auto-grading. Out of scope: full classroom management, advanced pedagogy analytics, large-scale user studies, and complex simulation backends.

1.6 Contributions

1. Requirements catalogue (functional/non-functional) for a focused block-learning MVP.
2. Architecture integrating Blockly, sandbox, graders, and content layer.
3. Extensible exercise JSON schema with localization and validation rules.
4. Prototype implementation and exemplar exercises demonstrating feasibility.

1.7 Thesis Structure

Chapter 2 surveys related systems and concepts. Chapter 3 summarizes the distilled requirements. Chapter 4 introduces the enabling technologies. Chapter 5 details the system architecture and data model. Chapter 6 outlines implementation highlights. Chapter 7 presents a technical evaluation. Chapter 8 discusses limitations. Chapter 9 concludes and sketches future work.

2 Related Work

This chapter reviews three thematic areas: (i) block-based programming environments (Scratch, Blockly derivatives, MakeCode), (ii) learning platforms with structured exercise flows (Brilliant-like micro-challenges), and (iii) approaches to automated grading and sandboxing in educational tooling. Prior work such as [**karle2017semantify**] illustrates structured semantic enrichment but differs in pedagogical scope. A comparative analysis motivates the design decisions in Chapter 5.

3 Requirements Summary

The full requirements engineering artefact (maintained in internal documentation) is distilled here.

3.1 Functional Overview

Core capabilities include: exercise model (metadata, toolbox, hints, tests), quiz assembly, authoring workflow (draft/publish, validation), client-side sandboxed execution, deterministic auto-grading (I/O normalisation, turtle command/state comparison), lightweight progress tracking, localization, and plugin-style extensibility for new blocks and exercise types.

3.2 Non-Functional Overview

Key non-functional goals: security (sandbox isolation, CSP), privacy (no external calls, optional anonymous mode), performance (sub-2s interaction latency), accessibility (WCAG-informed contrast and keyboard support), maintainability (modular layering, typed interfaces), and extensibility (stable plugin contracts).

3.3 Assumptions and Constraints

School network deployment implies intermittent connectivity and legacy browsers; thus dependencies are minimized and assets are locally hosted. Pedagogical depth evaluation is deferred.

4 Technologies

4.1 Blockly

Chosen over platform-style alternatives for fine-grained toolbox control and integration flexibility.

4.2 Web Stack

Prototype stack (planned): SvelteKit front-end, TypeScript for type safety, Tailwind for rapid UI iteration. Content persists as static JSON (upgradeable to SQLite).

4.3 Sandboxing

Browser iframe sandbox with restricted capabilities and message-passing protocol; loop-trap and timeout instrumentation enforce resource limits.

4.4 Grading Layer

Pure client graders for I/O and turtle tasks perform normalization, deterministic seeding, and structured result reporting.

4.5 Localization

JSON-based bilingual resource bundles; fallback chain ensures resilience.

5 System Design

This chapter formalizes the architecture: layered front-end, sandbox boundary, grading services, and content schema. Emphasis is placed on determinism, extensibility (block and exercise-type APIs), and privacy controls (feature flags, capability whitelists).

6 Implementation

Describes representative modules: exercise JSON validator, sandbox harness, turtle command log renderer, grader result model, and authoring form logic.

7 Evaluation

The prototype is evaluated along technical dimensions: (i) sandbox isolation effectiveness, (ii) grading determinism under repeated runs, (iii) performance metrics (load, execution latency), and (iv) expressiveness measured by coverage of targeted learning concepts in the curated exercises.

8 Limitations

Current scope excludes longitudinal learning outcome studies, large-scale classroom management, advanced accessibility adaptations (e.g. full screen reader optimization of Blockly), and server-side persistence in multi-user scenarios.

9 Conclusion and Future Work

The thesis delivered a technically grounded prototype and framework for extendable, privacy-preserving block-based learning. Future work includes richer exercise analytics, adaptive hint sequencing, expanded exercise types (state machines, physics), classroom orchestration features, deeper accessibility enhancements, and empirical pedagogical evaluation.