

**SETU Code Lab**

**Research Document**

Diarmuid O’Neill

South East Technological University

[Date of Submission]

Table of Contents

[Abstract 4](#_Toc211721484)

[Introduction 5](#_Toc211721485)

[Front-End 6](#_Toc211721486)

[React 6](#_Toc211721487)

[Angular 6](#_Toc211721488)

[Vue 6](#_Toc211721489)

[Django Template Language 6](#_Toc211721490)

[Blazor 6](#_Toc211721491)

[Blade 7](#_Toc211721492)

[Vite 7](#_Toc211721493)

[TypeScript 7](#_Toc211721494)

[SCSS 7](#_Toc211721495)

[Tailwind CSS 7](#_Toc211721496)

[Selected Front-End Technologies 8](#_Toc211721497)

[Back-End 9](#_Toc211721498)

[Node.js 9](#_Toc211721499)

[Express 9](#_Toc211721500)

[Django 9](#_Toc211721501)

[Flask 9](#_Toc211721502)

[FastAPI 9](#_Toc211721503)

[Laravel 10](#_Toc211721504)

[Selected Back-End Technologies 10](#_Toc211721505)

[Database 11](#_Toc211721506)

[SQL Database Options 11](#_Toc211721507)

[PostgreSQL 11](#_Toc211721508)

[MariaDB 11](#_Toc211721509)

[Supabase 11](#_Toc211721510)

[NoSQL Database Options 12](#_Toc211721511)

[MongoDB 12](#_Toc211721512)

[Firebase 12](#_Toc211721513)

[Selected Database 12](#_Toc211721514)

[Code Editors 13](#_Toc211721515)

[Monaco 13](#_Toc211721516)

[CodeMirror 14](#_Toc211721517)

[Ace 15](#_Toc211721518)

[Selected Code Editor 15](#_Toc211721519)

[Code Sandboxing 16](#_Toc211721520)

[Judge0 API 16](#_Toc211721521)

[Docker 16](#_Toc211721522)

[Selected Code Sandboxing Technology 17](#_Toc211721523)

[Testing 18](#_Toc211721524)

[Test Cases 18](#_Toc211721525)

[Implementation 19](#_Toc211721526)

[Benefits 19](#_Toc211721527)

[Gamification 20](#_Toc211721528)

[Similar Platforms 21](#_Toc211721529)

[Conclusion 22](#_Toc211721530)

[Appendix 23](#_Toc211721531)

[Glossary 24](#_Toc211721532)

[Bibliography 25](#_Toc211721533)

# Abstract

# Introduction

# Front-End

The front-end of SETU Code Lab needs to be clean, responsive and user-friendly so that users can focus on solving coding problems rather than learning how to use the platform. The chosen technology stack needs to support the latest two versions of Firefox, Google Chrome, and Microsoft Edge and remain readable and usable on the top five most popular desktop resolutions in Europe (1920x1080, 1536x864, 1366x768, 1280x720, 2560x1440) (Statcounter, 2025). Below are some of the technologies considered.

## React

React is an open-source JavaScript library developed by Meta (formerly Facebook) and first released in 2013 (GeeksforGeeks, 2025). It is used for building user interfaces and works by allowing the developer to create reusable user interface (UI) components. These components can then be put together to construct the full UI. React uses a virtual DOM (Document Object Model) to optimize rendering and improve performance by only updating the parts of the UI that have changed and not the whole page (Meta Platforms, Inc., 2025).

## Angular

Angular is full-featured, TypeScript based framework developed by Google and released in 2016, with a more opinionated architecture suitable for larger-scale, enterprise-level applications. It uses a component-based architecture like React and includes two-way-binding. This means that a change made to the UI can automatically update the component’s data model and vice versa (Google, 2025).

## Vue

Vue is another front-end framework developed by Evan You and first released in 2014. Vue was considered as a potential front-end technology for this project due to its simplicity, high performance and reactive two-way binding. This allows automatic synchronization of the UI and the underlying data (GeeksforGeeks, 2025).

## Django Template Language

Django is a full-stack web framework written in Python. This means it can be used for the front end and back end of the web application. It uses Django Template Language (DTL) to allow developers to generate HTML pages dynamically on the server and render them for users. This means that both the front end and back end could be written using the same framework (Django Software Foundation, 2025).

## Blazor

Blazor is a front-end web framework developed by Microsoft that uses HTML, CSS and C#. Using Blazor would allow a full .NET based technology stack. Blazor also allows developers to call into existing JavaScript libraries and APIs from C#. Blazor can run either on the server (Blazor Server) or directly in the browser via WebAssembly (Blazor WebAssembly) (Microsoft, 2024).

## Blade

Laravel is a full-stack web framework based on PHP. Laravel uses the Blade templating engine which generates HTML pages dynamically and supports reusable components and template inheritance. (Laravel, 2025).

## Vite

Vite is a modern front-end build tool and development server known for its fast speed, simplicity and support of modern browsers. It offers Hot Module Replacement (HMR) which updates the application instantly in the browser without needing a full page reload (Vite, 2024). This can be used with any of the previously mentioned frameworks and would be very useful for rapid development.

## TypeScript

TypeScript is a syntactic superset of JavaScript developed by Microsoft and first released in 2012. It extends JavaScript by adding static typing, which allows compile-time type checking. This means TypeScript will report any mismatched type errors before running the code whereas JavaScript will not. This is very helpful for debugging and helps improve the quality and performance of the code. (Microsoft, 2025).

## SCSS

SCSS is a stylesheet language that is compiled into CSS. SCSS syntax is fully compatible with CSS and includes more advanced features such as variables, nested rules, mixins, and built-in modules. SCSS variables differ from CSS variables in that CSS variables have different values for different elements, whereas SCSS variables remain the same across multiple elements. This is very useful for reducing code repetition. Nested rules further reduce code repetition by letting an inner rule inherit selectors from an outer rule. Mixins are reusable blocks of CSS that can be defined and used throughout the stylesheet mixed in with other styles. The Built-in modules feature provides functions that are useful for manipulating numbers, strings, colours and more, making it easier to build dynamic stylesheets (SASS Team, 2025).

## Tailwind CSS

Tailwind CSS uses utility classes for rapid development. Tailwind utility classes allow developers to combine many single purpose utility classes directly in the markup i.e. not in a dedicated CSS file. This allows changes to be made faster as the developer does not have to consider what to name their classes and how their changes will affect other pages. Tailwind is also efficient as it purges any unused CSS, reducing the final bundle size (Tailwind Labs, 2025).

One issue with Tailwind CSS is that its utility first approach makes readability and maintainability more difficult as all styles are applied in an inline fashion. This bloats the codebase and becomes a headache for the developer. Another issue with Tailwind is that it’s utility first design and lack of dedicated stylesheets is unfamiliar to new developers.

## Selected Front-End Technologies

The chosen technologies for the front-end are React, Vite, TypeScript, and SCSS.

React was chosen as the front-end framework for SETU Code Lab due to its unidirectional data flow which simplifies debugging and state management. The MobX state management library allows for the implementation of two-way binding if needed (MobX, 2025). React also allows for increased interactivity as opposed to Django template language, Blazor and Blade. With React, the user interface can update in real time without needing a full page reload. It is also more flexible and integrates more naturally with the chosen technology stack, particularly TypeScript which provides strong static typing and full compatibility with JSX the syntax extension used by React (Microsoft, 2025).

Vite has been chosen as a build tool for use with React to speed up development and for its native support of the latest versions of modern browsers. SCSS has been chosen ahead of Tailwind CSS because of its better readability, maintainability, additional features and familiar design methodology.

# Back-End

The back end of SETU Code Lab is responsible for all server-side operations and data processing. It will implement CRUD (Create, Retrieve, Update, Delete) functionality for lots of different types of data such as, code problems, test cases, profiles, submissions and metrics. It will also handle user authentication, allowing role-based access (e.g. Student and Lecturer) and integrate with the chosen code sandboxing technology to ensure user-submitted code is executed safely.

## Node.js

Node.js is an open-source JavaScript runtime environment. It allows developers to run JavaScript outside of the web browser. It is asynchronous and event-driven meaning it can handle multiple tasks at once without blocking others, making it very efficient. It is built on Google’s V8 JavaScript engine, which compiles JavaScript into machine code, resulting in a high level of performance. It’s large ecosystem of packages available through npm (Node Package Manager) also make it a popular choice for backend development (OpenJS Foundation, n.d.).

## Express

Express is the most popular web framework for Node.js. It is unopinionated, lightweight and fast. It provides features such as routing, HTTP helpers, middleware functions and middleware error handling. It is also used for building RESTful APIs and simplifies database interactions (W3Schools, n.d.).

## Django

As mentioned in the front-end technologies section, Django is a full-stack web framework written in Python. Django offers many built-in features like authentication and authorization tools, Object Relational Mapping (ORM) and an admin interface (GeeksforGeeks, 2025).

## Flask

Flask is a lightweight Python micro web framework. This means unlike Django it is more unopinionated and flexible about how features are implemented. However, it does include features such as URL routing and the Jinja2 template engine which make routing and developing a front end easier. Features such as authentication and authorization are not included by default and require extensions (Flask-Login and SQLAlchemy) (GeeksforGeeks, 2025) (Pallets Projects, 2025).

## FastAPI

FastAPI is a fast web framework, used for building APIs with Python. As its name suggests, FastAPI has very high performance claiming to be on par with Node.js. It is fully compatible with the Pydantic library allowing custom data types and validation. It also supports automatic API documentation generation using either Swagger UI or ReDoc (Ramírez, n.d.).

## Laravel

Laravel is a full-stack web framework based on PHP. It is highly opinionated and features built-in authentication and authorization tools, URL routing and generation, an ORM making it easy to interact with the database and uses the Blade templating engine to dynamically generate HTML pages. Laravel also includes a Command Line Interface (CLI) called Artisan. This gives developers useful commands that they can use to automate common tasks such as generating boilerplate code, running tests, and performing database migrations (Laravel, 2025).

## Selected Back-End Technologies

Node.js and Express have been chosen as the back-end technologies for SETU Code Lab. This is due to their seamless integration with React and TypeScript, high performance, large number of useful libraries available through npm, and their asynchronous, event driven architecture. This should allow for rapid, high-quality development and give the developer fine-tuned control over how things are implemented.

# Database

SQL (or relational) databases hold structured data and NoSQL (non-relational) databases can hold semi-structured and unstructured data. For example, an SQL database will store data in tables containing specific rows and columns whereas a NoSQL database can store data differently depending on what it is. SQL databases also typically scale vertically (e.g. one database scales up) and NoSQL databases typically scale horizontally (e.g. the number of databases increases) (MongoDB, n.d.).

Choosing the right database for SETU Code Lab depends on the kind of data that needs to be stored. The system will likely store mainly structured data (users, problems, results, etc.) with the possibility of some semi-structured data likely in JSON format (submission logs, test cases).

## SQL Database Options

### PostgreSQL

PostgreSQL is a relational (SQL) database management system known for its reliability, performance and robust features. With nearly forty years of active development PostgreSQL offers advanced features such as full-text search, JSON and JSONB support and custom data types (PostgreSQL, 2025).

Instead of searching for exact strings, full-text search can find results that semantically match. For example, if a student entered a search containing the word “loop”, PostgreSQL can return any problems whose titles or descriptions contain the word “loop” or similar words such as “loops” or “looping”. Support for JSON and JSONB is useful as the system may need to store some semi-structured data such as problem test cases or submission logs. Support for custom data types is also needed for SETU Code Lab as there will be many custom data types such as difficulty level (e.g. easy, medium, hard) (PostgreSQL, 2025).

### MariaDB

MariaDB is a relational (SQL) database management system developed by the original creators of MySQL. Like PostgreSQL it is fully ACID (Atomicity, Consistency, Isolation and Durability) compliant and highly compatible with MySQL. MariaDB supports dynamic columns, a feature allowing developers to store multiple columns in a single row in a database. This is useful for storing data with many different attributes however it does not support fully custom data types like PostgreSQL (MariaDB Foundation, n.d.).

### Supabase

Supabase is an open-source Back End as a Service (BaaS) platform founded in 2020. It provides many features for server-side development such as authentication, auto-generated APIs, edge functions and a full PostgreSQL database (Ayezabu, 2022).

An advantage of a BaaS technology is the simplification and speeding up of back-end development and database set up. However, some applications require custom functionality that is not provided by these services (Ayezabu, 2022). In this case Supabase does not provide functionality for code sandboxing or the ability to spin up docker containers.

## NoSQL Database Options

### MongoDB

MongoDB is a non-relational (NoSQL) database management system. It allows developers to store non-relational data such as JSON-like documents which can better reflect how the stored objects are used in code. This could prove useful in case of changing requirements. Other features listed on the MongoDB website include consistency with ACID transactions, built-in querying capabilities and serverless horizontal scaling which may assist future scaling of the application (MongoDB, n.d.).

### Firebase

Firebase was founded in 2011 and acquired by google in 2014. It is a Back End as a Service (BaaS) platform which features services like hosting, authentication, cloud functions, cloud messaging, A/B testing and two types of non-relational (NoSQL) databases.

The first database is called the Firebase Realtime Database. It is a cloud-hosted database that stores JSON data and can synchronize for all users in real time. The other database service available with Firebase is the Cloud Firestore. This database is like the real time database; however, it is a document-oriented NoSQL database that stores data in collections of documents (Ayezabu, 2022).

## Selected Database

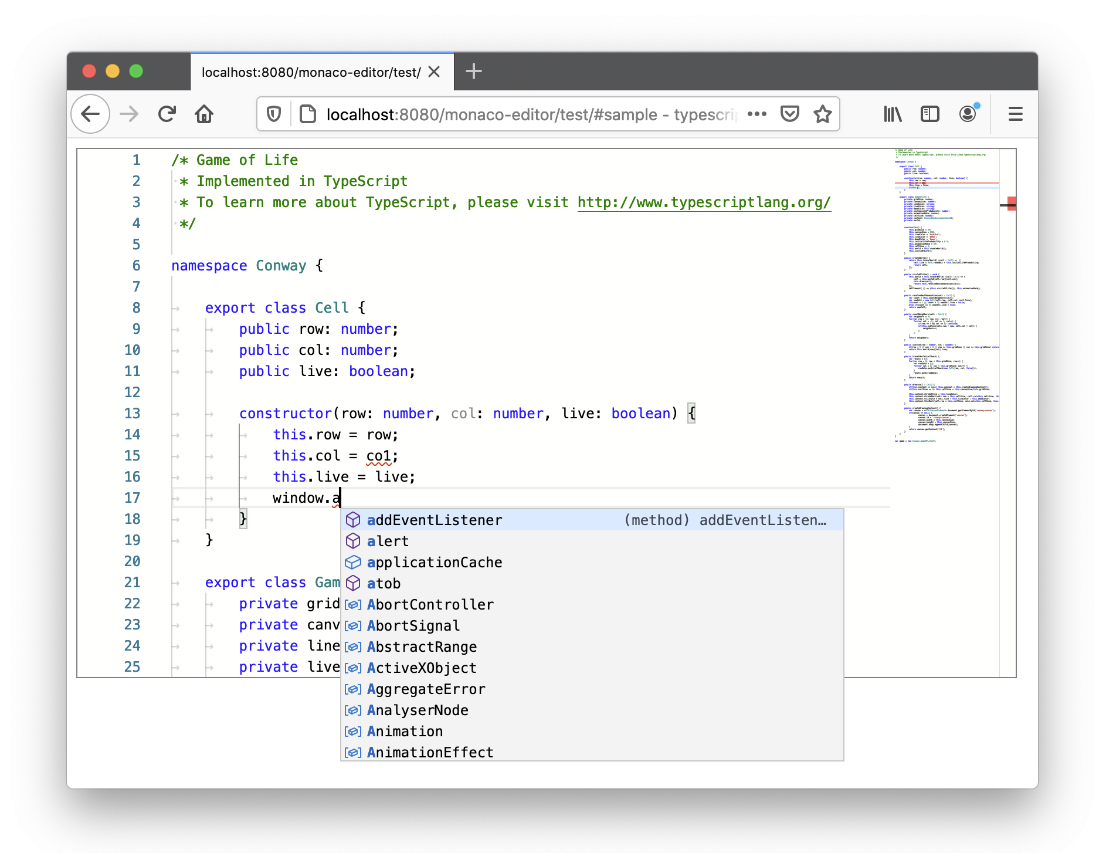
PostgreSQL has been selected as the database to support SETU Code Lab. This is because it can store relational data unlike MongoDB. Custom data types and semi structured data in JSON and JSONB format are also supported, something that is not available in MariaDB. The full-text search feature will also be useful for allowing users to search for specific problems, support for role management will help to implement different levels of access (e.g. Student and Lecturer) and the seamless Node.js integration provides a simplified developer experience.

# Code Editors

SETU Code Lab aims to have a built-in code editor in the browser in which the user can input their solution code. This editor needs to be fast, reliable and user friendly. Several libraries have been identified for this task such as Monaco, CodeMirror and Ace. These are all available through npm and compatible with the rest of the chosen technology stack.

## Monaco

Monaco is the code editor used by Visual Studio Code and thus has the same look and feel. It is written in TypeScript, and the latest version is version 0.54.0. It has many of the same features as Visual Studio Code such as IntelliSense meaning it has a rather large bundle size. IntelliSense is Visual Studio Codes signature code completion, content assist, and code hinting tool. This functionality can be enabled or disabled as needed and supports almost any programming language (Microsoft, n.d.).



**Fig. X.** Monaco editor example **Source:** (Microsoft, n.d.)

## CodeMirror

CodeMirror is another feature rich code editor though it has a much smaller bundle size compared to Monaco (153kb vs 21.3kb) (NPM, 2025) (NPM, 2025). It is written in JavaScript, and the latest version is CodeMirror 6. Some of the potentially useful features include syntax highlighting which colours code to reflect its syntactic structure, language specific autocompletion hints, accessibility support for screen readers and keyboard only users and undo and redo functionality. CodeMirror also remains responsive even for very large documents (Haverbeke, n.d.).

A screenshot of a computer

AI-generated content may be incorrect.

**Fig. X** CodeMirror 6 editor example **Source:** (CodeMirror, 2025)

## Ace

The Ace code editor is another lightweight, embeddable code editor like CodeMirror. It has many of the same features such as syntax highlighting, code folding, themes and customizable key bindings. It also includes basic autocompletion and search and replace functionality (Ace, 2024). While Ace provides good language support and performance, its architecture is older and less modular compared to CodeMirror or Monaco. This means it can be more difficult to integrate with React (Masad, 2025).

A screenshot of a computer

AI-generated content may be incorrect.

**Fig. X** Ace code editor example **Source:** (Ajax.org, 2025)

## Selected Code Editor

CodeMirror 6 has been selected as the code editor for SETU Code Lab as it strikes the right balance between performance, due to its small bundle size, functionality, which is extensible if needed and ease of integration with front-end frameworks such as React.

# Code Sandboxing

When a user of SETU Code Lab runs or submits a solution to a code problem, it must be run safely in a containerized environment. This is to prevent any potentially malicious code from causing harm. Running code inside a docker container for instance allows developers to: limit the amount of system resources available, preventing any Denial of Service (DOS) type attacks and disable outbound networking, preventing any network abuse or exfiltration of data. Another benefit of running submitted code in a containerized environment is that every student’s submission will be run under the exact same conditions ensuring fairness and reliability.

## Judge0 API

Judge0 is an open source, online code execution system. It provides detailed API documentation, a highly scalable architecture, support for over sixty programming languages, detailed execution results and HTTP callbacks (Došilović, 2024). The service is free under the GPL-3.0 licence for self-hosted applications, but various paid plans are available for other hosting options (Došilović, 2025).

Judge0 works by taking in a language id (this specifies which programming language is being inputted) and the source code you wish to run via its API. This code is then run in a sandboxed environment with limited resources, and the results are returned. There is also support for multi-file programs (Došilović, 2024).

## Docker

Docker is a platform for developing and running applications inside isolated environments. These isolated environments are called containers. Developers can give containers a limited amount of system resources and install the needed dependencies to allow an application, or in the case of SETU Code Lab, allow users’ code submissions to run safely. This ensures that every code submission runs under the exact same conditions, restricts network access and prevents any potential security issues (Docker, inc., 2024).

Docker Engine is the technology that would be used by SETU Code Lab to create containers. When a user makes a submission, the backend communicates with the Docker Engine API to create a new container. This container is based on a predefined image that includes the required dependencies to run the submitted code. An image is like a blueprint which tells Docker how to set up the container and what needs to be installed inside of it (GeeksforGeeks, 2025). For example, the eclipse-temurin docker image provides a Java Development Kit (JDK) which allows Java code to execute inside a container (Eclipse Foundation, 2025).

## Selected Code Sandboxing Technology

Docker has been chosen as the code sandboxing technology for SETU Code Lab. While the Judge0 API offers a quick and easy solution for code sandboxing, it lacks the precise control over resource limits that Docker provides.

A Judge0 implementation lacks the technical depth required for a final year project and supports a limited amount of programming languages (60+), whereas a custom Docker solution could theoretically support any language and its libraries. Finally, a fully Docker based solution will allow the developer to decide exactly how submissions are run and what format the results are returned in. This can help optimize performance, maintenance and promote rapid development.

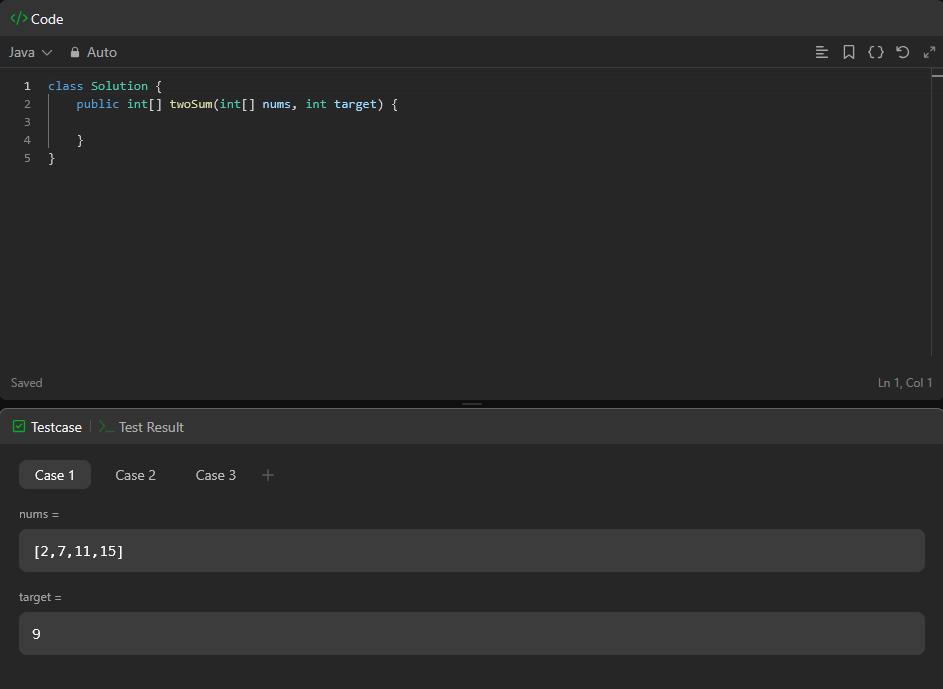
# Testing

When a student submits a solution to a particular code problem, the system must have some way of determining if the code is correct or not. This is where automated code testing comes in. Testing is the process of ensuring software works as intended by observing its execution and comparing actual results with expected results. Unit or component testing focuses on testing the smallest testable parts of code and can easily be automated (Dorothy, et al., 2008).

## Test Cases

A test case contains a test input value and an expected output value (Dorothy, et al., 2008). A similar platform called HackerRank uses two types of test cases for testing submissions. Sample Test Cases are visible to the user and helps them to understand the problem, and Hidden Test Cases are not visible to the user and are there to prevent users from hard coding expected outputs. Hidden Test Cases encourage users to write more robust code and generalized solutions. HackerRank recommends 2-3 sample test cases to help users grasp the problem and 8-15 test cases in total to ensure a full evaluation (HackerRank, 2025).

When creating a problem, HackerRank also allows users to define skeleton functions with specific return types and parameters. This way, test inputs and test outputs will match up exactly with the parameters and return types defined in the empty function (HackerRank, 2025). Another similar platform called LeetCode also does this.



**Figure X.** Skeleton code from LeetCode **Source:** (LeetCode, 2021)

## Implementation

SETU Code Lab will store several test cases in the database for each code problem. Multiple tests are needed to ensure that submissions are fully correct and robust. When the user attempts to run or submit their code, the test input values are passed into the function the student has just written. This function is then executed inside an isolated docker container. After execution the return value is sent back to the backend to be judged, and the Docker container is destroyed to stop it consuming resources.

The return value is then compared against the expected output value. If the actual output value matches the expected output value the test is considered a pass. If all test cases pass the student has solved the code problem correctly. If none or only some test cases pass, the student has not solved the code problem, and this is considered a failure.

## Benefits

This testing approach eliminates human error in grading which improves consistency and provides instant feedback to students improving efficiency. It does require however, that when creating new code problems and test cases, Lecturers input correctly formatted skeleton functions and test cases.

# Gamification

# Similar Platforms

# Conclusion

# Appendix

# Glossary

# Bibliography

Ace, 2024. *Ace - The High Perfromance Code Editor for the Web.* [Online]   
Available at: https://ace.c9.io/  
[Accessed 13 October 2025].

Django Software Foundation, 2025. *Templates | Django Docs.* [Online]   
Available at: https://docs.djangoproject.com/en/5.2/topics/templates/  
[Accessed 15 October 2025].

GeeksforGeeks, 2025. *Difference Between React.js and Angular.js.* [Online]   
Available at: https://www.geeksforgeeks.org/reactjs/difference-between-react-js-and-angular-js/  
[Accessed 8 10 2025].

GeeksforGeeks, 2025. *Flask vs Django.* [Online]   
Available at: https://www.geeksforgeeks.org/python/differences-between-django-vs-flask/  
[Accessed 15 October 2025].

GeeksforGeeks, 2025. *History and Evolution of React.* [Online]   
Available at: https://www.geeksforgeeks.org/reactjs/history-and-evolution-of-react/  
[Accessed 8 October 2025].

GeeksforGeeks, 2025. *Vue.js Tutorial - GeeksforGeeks.* [Online]   
Available at: https://www.geeksforgeeks.org/javascript/vue-js/  
[Accessed 8 October 2025].

Google, 2025. *What is Angular?.* [Online]   
Available at: https://angular.dev/overview  
[Accessed 8 10 2025].

Haverbeke, M., n.d. *CodeMirror Documentation.* [Online]   
Available at: https://codemirror.net/  
[Accessed 13 October 2025].

Laravel, 2025. *Blade Templates | Laravel Docs.* [Online]   
Available at: https://laravel.com/docs/12.x/blade  
[Accessed 15 October 2025].

Laravel, 2025. *Laravel Docs.* [Online]   
Available at: https://laravel.com/docs/12.x  
[Accessed 15 October 2025].

Masad, F., 2025. *Replit - Comparing Code Editors.* [Online]   
Available at: https://blog.replit.com/code-editors  
[Accessed 13 October 2025].

Meta Platforms, Inc., 2025. *Quick Start - React.* [Online]   
Available at: https://react.dev/learn  
[Accessed 8 October 2025].

Microsoft, 2024. *Blazor Docs.* [Online]   
Available at: https://learn.microsoft.com/en-us/aspnet/core/blazor/?view=aspnetcore-9.0&WT.mc\_id=dotnet-35129-website  
[Accessed 15 October 2025].

Microsoft, 2025. *TypeScript Documentation.* [Online]   
Available at: https://www.typescriptlang.org/docs/  
[Accessed 9 October 2025].

Microsoft, 2025. *TypeScript Documentation - JSX.* [Online]   
Available at: https://www.typescriptlang.org/docs/handbook/jsx.html  
[Accessed 8 October 2025].

Microsoft, n.d. *IntelliSense in Visual Studio Code.* [Online]   
Available at: https://code.visualstudio.com/docs/editing/intellisense  
[Accessed 12 October 2025].

MobX, 2025. *MobX Documentation.* [Online]   
Available at: https://mobx.js.org/README.html  
[Accessed 8 October 2025].

MongoDB, n.d. *MongoDB Documentation | Homepage.* [Online]   
Available at: https://www.mongodb.com/docs/  
[Accessed 15 October 2025].

MongoDB, n.d. *NoSQL VS SQL Databases | MongoDB.* [Online]   
Available at: https://www.mongodb.com/resources/basics/databases/nosql-explained/nosql-vs-sql  
[Accessed 15 October 2025].

NPM, 2025. *Npm - CodeMirror V6.* [Online]   
Available at: https://www.npmjs.com/package/codemirror  
[Accessed 13 October 2025].

NPM, 2025. *Npm - Monaco Editor React.* [Online]   
Available at: https://www.npmjs.com/package/@monaco-editor/react  
[Accessed 13 October 2025].

OpenJS Foundation, n.d. *Node.js - Introduction to Node.js.* [Online]   
Available at: https://nodejs.org/en/learn/getting-started/introduction-to-nodejs  
[Accessed 11 10 2025].

Pallets Projects, 2025. *Flask Docs.* [Online]   
Available at: https://flask.palletsprojects.com/en/stable/design/#the-routing-system  
[Accessed 15 October 2025].

PostgreSQL, 2025. *PostgreSQL: Documentation.* [Online]   
Available at: https://www.postgresql.org/docs/current/  
[Accessed 11 October 2025].

Ramírez, S., n.d. *Features | FastAPI.* [Online]   
Available at: https://fastapi.tiangolo.com/features/  
[Accessed 15 October 2025].

SASS Team, 2025. *SASS Documentation.* [Online]   
Available at: https://sass-lang.com/documentation/  
[Accessed 9 October 2025].

Statcounter, 2025. *Desktop Screen Resolution Stats Europe.* [Online]   
Available at: https://gs.statcounter.com/screen-resolution-stats/desktop/europe  
[Accessed 4 October 2025].

Tailwind Labs, 2025. *Getting Started with Tailwind CSS.* [Online]   
Available at: https://tailwindcss.com/docs/installation/using-vite  
[Accessed 10 October 2025].

Vite, 2024. *Getting Started | Vite.* [Online]   
Available at: https://vite.dev/guide/  
[Accessed 8 October 2025].