

Emerald: Online Guideline Checking Tool

Patrick Berchtold, Nico Gabriel, Martin Rabensteiner, and Marcel Zisser

706.041 Information Architecture and Web Usability 3VU WS 2024/2025
Graz University of Technology

05 Feb 2025

Abstract

This project report describes Emerald, an open-source web application for guideline checking. This tool enables the creation and management of reviewing projects, and also the review process itself. Emerald is built using Node.js for the backend, and Angular for the frontend. It is organised as an Nx monorepo for ease of development and deployment. The responsive frontend enables users to work with Emerald on both mobile and desktop devices. The database connection is handled with Prisma, which decouples the application from using a particular database system. This also simplifies the hosting of an Emerald instance. Functionally, the app supports three different user roles: admins for user management, project owners for project creation and management, and reviewers for the review process. This clean separation makes the use of Emerald as easy and straightforward as possible for its users. Proper authentication and authorisation procedures for security are also implemented.

© Copyright 2025 by the author(s), except as otherwise noted.

This work is placed under a Creative Commons Attribution 4.0 International (CC BY 4.0) licence.

Contents

Contents	i
List of Figures	iii
1 Introduction	1
2 Implementation	3
2.1 Database	3
2.2 Backend	3
2.3 Frontend.	3
2.4 Authentication and Authorisation.	4
2.5 Dashboard	4
2.6 Project Creation	5
2.7 Reviewing	5
2.8 Project Progress	6
3 Deployment	7
4 Future Work	9
Bibliography	11

List of Figures

2.1	Emerald: User Management	4
2.2	Emerald: Project Owner Dashboard	5
2.3	Emerald: Reviewer Dashboard	5
2.4	Emerald: Criterion Creation	6

Chapter 1

Introduction

Emerald is an open-source web application to simplify and facilitate guideline checking [Berchtold et al. 2025]. This includes creating, managing, and reviewing checklists (projects). Emerald delivers a modern, lightweight, open-source solution under an MIT license [MIT 2025]. The implementation was done in a limited time of around six weeks and is therefore not completely mature, but is stable and is ready to be extended or improved in the future. The project was part of the course Information Architecture and Web Usability, taught by Keith Andrews at Graz University of Technology in the winter semester 2024/25 [Andrews 2025].

Testpad is a commercial guideline checking tool [Testpad 2025], which serves as inspiration for some parts of Emerald. It includes criteria grouping, review inspection, PDF reports, comments on criteria, and third-party integrations such as bug tracking. However, Testpad only supports binary criteria, there are no point-based criteria. Also, Testpad does not provide a free version, nor any option for self hosting. The paid version of Testpad is limited to 25 reviewers at most, and costs \$10 per month per reviewer. Testpad offers a free trial for 30 days.

In 2023, a previous group already implemented a version of Emerald [Aumüller et al. 2024]. It was decided not to continue this work, but to implement a new version from scratch, instead of trying to understand and then modify the old code. A dependency on Netlify was also problematic, due to the usage of Netlify functions, which means that no proper backend existed.

In order to have consistent language and wording across the application, some terminology was defined. For the data model, the following entities were defined:

- *Criterion*: A criterion is a single entity that has to be checked during a review process. A criterion can be either binary (pass/fail) or point-based.
- *Project*: A project is a set of criteria. For clarity, criteria are grouped into criteria groups.
- *Review*: A review is produced by a reviewer going through the criteria of a project and evaluating them.

In addition, three types of user were defined:

- *Admin*: An admin user can only add, edit, and delete users. An admin cannot create or maintain projects or perform reviews.
- *Project Owner*: A project owner can create a project, specify its criteria, assign reviewers to it, and see the results. Project owners can also be assigned to review projects themselves.
- *Reviewer*: A reviewer can only perform reviews. Reviewers do not have any other permissions.

Chapter 2

Implementation

This chapter describes how Emerald was implemented. The software stack was selected according to multiple criteria. The application should be able to be self-hosted and not depend on third-party cloud services.

2.1 Database

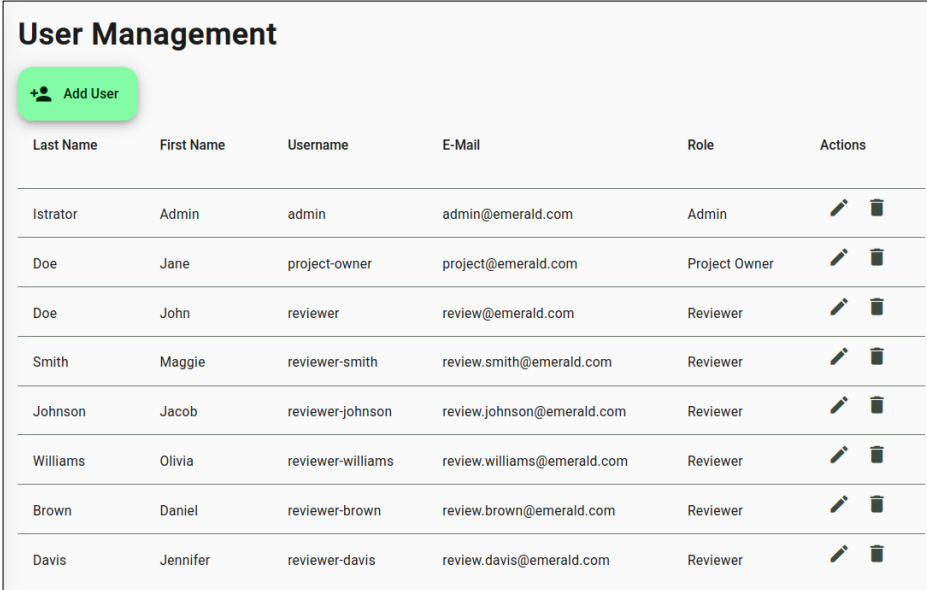
During development, a PostgreSQL database was used to persist the application data. Within the project repository, a docker-compose file sets up a containerised database for testing purposes. In production, Emerald can be paired with a many different databases, because Prisma [Prisma 2025] is used as an ORM. This allows switching between different database systems with relative ease, just by tweaking of the Prisma configuration file.

2.2 Backend

Relying on any cloud-based solution for the backend was not an option, therefore an own backend was created using Nest.js [Mysliwiec 2025]. Nest.js allows the implementation of efficient and scalable Node.js server-side applications. It is built with TypeScript, which perfectly pairs with Angular, the frontend framework used. Under the hood, Nest.js uses Express.js, which is a robust HTTP server framework. Nest.js offers the functionality to statically serve a frontend application. This means, that only one port to deploy both backend and frontend is needed. It is only necessary to build the frontend into the corresponding directory and it will be started together with the backend.

2.3 Frontend

For the frontend framework, the decision was to use Angular [Google 2025]. Angular is a frontend framework developed and maintained by Google, which allows developers to build fast and reliable applications. The frontend is structured into libraries. There are several feature libraries, which have a specific purpose and are not designed to be reusable. An example for a feature library would be the dashboard library, in which the components for the dashboard view are implemented. For reusable components, several shared library exists. Shared libraries contain components and parts of the project, which are used by multiple features. Often they are also shared across projects, which means that also the backend uses components from the shared libraries, such as the data models.



















Last Name	First Name	Username	E-Mail	Role	Actions
Istrator	Admin	admin	admin@emerald.com	Admin	 
Doe	Jane	project-owner	project@emerald.com	Project Owner	 
Doe	John	reviewer	review@emerald.com	Reviewer	 
Smith	Maggie	reviewer-smith	review.smith@emerald.com	Reviewer	 
Johnson	Jacob	reviewer-johnson	review.johnson@emerald.com	Reviewer	 
Williams	Olivia	reviewer-williams	review.williams@emerald.com	Reviewer	 
Brown	Daniel	reviewer-brown	review.brown@emerald.com	Reviewer	 
Davis	Jennifer	reviewer-davis	review.davis@emerald.com	Reviewer	 

Figure 2.1: Emerald: User management interface as seen by an admin.

2.4 Authentication and Authorisation

A user has several properties, including first name, surname, role, and email, as well as username and password. The username and password are needed for authentication. User management is facilitated by an admin, as shown in Figure 2.1.

After a user is successfully authenticated, a JSON Web Token (JWT) is set in a cookie to authenticate the user for the session. The JWT is valid for five minutes, before the client needs to re-authenticate. To not make this annoying for the user, token refresh logic is in place, which retrieves a new JWT as soon as the old token is invalid. For this to work, there is a client side interceptor of all responses, which checks if the backend answered with the status code “401 Unauthorized”. In that case, a request is sent to a special refresh endpoint, which checks if a valid refresh JWT is set as an HTTP-only cookie. This special JWT is generated when the user logs in and cannot be read or changes from the client. This ensures it that the cookie is not hijacked by an unauthorized user. Once the user logs out, both the regular JWT as well as the refresh JWT are deleted server-side.

Both frontend and backend implement a role-based authorisation concept. The role of the user is retrieved from the backend upon authentication and can be read by the client from the retrieved JWT token. Depending on the role, different features of Emerald are available to the user. The backend is very similar, as each route is protected by a guard, which checks if the requesting user is authorised to activate it.

2.5 Dashboard

The dashboard is the start screen for each user. Depending on the type of user, different information is displayed. Currently, only the dashboards for project owners and reviewers are implemented. A project owner sees a summary of their own projects, followed by any reviews they have been assigned, as shown in Figure 2.2. A reviewer see an overview of their assigned reviews, as shown in Figure 2.3.

Dashboard

My Projects

Here is an overview of your most recently created projects. To get a all your projects, please [click here](#)

Checklist Homepage Design 0 out of 6 reviews completed ^

The Web Usability Checklist. The checklist covers 46 individual guidelines.

Reviews ● Passed ● Failed ● Pending

Reviewer	Assignment Date	Due Date	Review Status	Criteria Summary
Doe John	2025-01-28	2025-01-31	In Progress	<div style="width: 50%; background-color: #ccc; height: 10px; margin-bottom: 5px;"></div> 46 ↗
Smith Maggie	2025-01-28	2025-01-31	Not Started	<div style="width: 0%; background-color: #ccc; height: 10px; margin-bottom: 5px;"></div> 46 ↗
Johnson Jacob	2025-01-28	2025-01-31	Not Started	<div style="width: 0%; background-color: #ccc; height: 10px; margin-bottom: 5px;"></div> 46 ↗

Figure 2.2: Emerald: Dashboard as seen by a project owner.

Dashboard

My Reviews

Here is an overview of your most recently assigned reviews. To get a all your checklists, please [click here](#)

● Passed ● Failed ● Pending

Project Title	Project Owner	Assignment Date	Due Date	Review Status	Criteria Summary
Checklist Homepage Design	Jane Doe	2025-01-28	2025-01-31	In Progress	<div style="width: 50%; background-color: #ccc; height: 10px; margin-bottom: 5px;"></div> 46 ↗

Figure 2.3: Emerald: Dashboard as seen by a reviewer.

2.6 Project Creation

Project creation consists of three steps: general settings, criteria definition, and reviewer assignment. As part of the general settings, a title, a description, and a due date are specified. Criteria are defined in the second step, shown in Figure 2.4. Criteria are grouped, and every group is given a title and optional description. Each criterion is either binary or point-based. For a binary criterion, a reviewer can only assign a pass or fail status in the review process. For a point-based criterion, a reviewer can assign points in a specified range. The maximum points amount are set by the project owner on a per criterion basis. The third and final step of project creation is to assign reviewers to the project.

2.7 Reviewing

Once a reviewer has been assigned to a project, the reviewer can access the project to enter a review. For binary criteria, the user can select Pass or Fail. For point-based criteria, the reviewer can enter a positive number upto the maximum points for the criterion. In either case, choices can be reverted or left blank, so the criterion is flagged as Pending. Additionally, reviewers have the option to add comments for each criterion.

Define Criteria

Define the criteria groups and criteria for the project.

Screenshots

Group Name*
Screenshots

Group Description

Criteria

Criterion creation screenshot is in the report.

Criterion Description
Criterion creation screenshot is in the report.

Binary Points

Delete Criterion

Add Criterion Delete Criteria Group

Figure 2.4: Emerald: Creating a new binary criterion.

2.8 Project Progress

The project owner can monitor the project's progress by selecting any number of reviewers. They are displayed in an overview table, where the criteria are listed as rows, and each reviewer's results are displayed in separate columns.

Chapter 3

Deployment

The Emerald source code can be found on GitHub [Berchtold et al. 2025]. To deploy the application, the server must be capable of running Node.js. At the time of creation, the current LTS version of Node.js is 22.13.1 and Emerald supports this Node version. It is very possible that reasonably older or newer versions also work, but this cannot be guaranteed.

Emerald requires a database to store its data. The database stores both users and projects. Since Prisma is used for the database connection, the database provider does not matter, but the default is PostgreSQL. If another provider is used, this needs to be changed in the code, but this is only a change in a single line of code which is described in the project's `README.md` file. The configuration parameters for the database connection (host, username, etc.) are provided with an `.env` file.

To build the application from the source code, the following instructions are necessary:

1. `npm install`

2. `npm run build:production`

3. Optional: `npm run generate-demo-data`

Caution: This command will completely reset the database defined in the `.env` file, apply the correct database schema for Emerald and populate the database with demo data. Make sure to create database backup before doing that.

4. Once Emerald has been built, the Node.js server is launched by using the command:

```
node ./dist/apps/emerald-api/main.js
```

from the root directory of the repository.

Chapter 4

Future Work

Not every possible or useful feature was implemented, due to the short development timeframe of the project. The implementation process instead focused on the core features defined in the project discussion sessions. Many improvements and new features could be added to the application in the future:

- *Project Editing*: Once a project is created, it cannot be edited. Neither the metadata, criteria, nor reviewers can be edited. This could be implemented by duplicating the creation component and making a few changes in the loading and saving process.
- *Password Management*: In the current state, when creating a new user, an admin has to set a password and send it to the user. This is not very secure, since it has to be transmitted in plain text in order to use it. Also, the user is then not forced to change it after his first login. There is also currently no logic for resetting an existing password.
- *Results and Analytics*: Currently, results are only available by criterion. There is an overview of all reviews and criteria within a project as a table, but the results are not summed up or analysed in any way. To enhance the visualization and analysis of review results, a chart library could be used to produce summary statistics.
- *Import and Export*: Being able to export and import an entire project as a .json file, including reviews and results, would deliver a practical way to support the transfer between multiple Emerald instances. It would also be beneficial to be able to save and load a set of criteria in the form of a .csv file.
- *Save as PDF*: Saving a project to PDF might be helpful in the review workflow. For example, an empty criteria sheet could be printed out to do an offline review. Or various reports could be generated of the results of a project.
- *Bug Tracking*: Bug tracking functionality could be implemented by integrating external tools like Jira, allowing for seamless issue management and tracking within a review. This integration could also feature automatic ticket creation when bugs are identified.
- *User Testing*: The system would benefit from some formative user testing, such as a thinking aloud test, of the admin, project creation, and review processes.

Bibliography

- Andrews, Keith [2025]. *Information Architecture and Web Usability*. 31 Jan 2025. <https://courses.isds.tugraz.at/iaweb> (cited on page 1).
- Aumüller, Thomas, Daniel Liegl and Fabian Platzer [2024]. *Emerald: A Web-Based Guideline Checking Tool*. 04 Feb 2024. <https://github.com/danielliegl/emerald> (cited on page 1).
- Berchtold, Patrick, Nico Gabriel, Martin Rabensteiner and Marcel Zisser [2025]. *Emerald: Online Guideline Checking Tool*. 31 Jan 2025. <https://github.com/marcel-zisser/emerald> (cited on pages 1, 7).
- Google [2025]. *Angular*. 31 Jan 2025. <https://angular.dev/> (cited on page 3).
- MIT [2025]. *Exploring the MIT Open Source License: A Comprehensive Guide*. 29 Jan 2025. <https://tlo.mit.edu/understand-ip/exploring-mit-open-source-license-comprehensive-guide> (cited on page 1).
- Mysliwicz, Kamil [2025]. *NestJS: A Progressive Node.js Framework*. 31 Jan 2025. <https://nestjs.com/> (cited on page 3).
- Prisma [2025]. *Prisma: Simplify Working and Interacting with Databases*. Prisma Data, 31 Jan 2025. <https://prisma.io/> (cited on page 3).
- Testpad [2025]. *Testpad: A Test Plan Tool for Simpler Test Case Management*. 31 Jan 2025. <https://testpad.com/> (cited on page 1).