Design and Implementation of a Conference Management System for Use in Academic Institutions

Soares, J.[1] and Carvalho, P.[1,2]
[1] ISEP-PPorto, Porto, Portugal
[2] GILT, ISEP-PPorto, Porto, Portugal

Abstract. This paper presents the development of a web-based conference management system designed to serve the research units of universities. The system supports the creation and management of multiple academic conferences, providing functionalities for paper submission, peer review, and event scheduling while ensuring role-based access control and customization.
The implementation used SvelteKit, TailwindCSS, and MongoDB, leveraging modern web frameworks to enhance user experience and scalability. Although not all features were completed, notably the customizable forms and automatic program generation, the project achieved the main functional objectives and provided valuable insights into contemporary full-stack web development.

Keywords: Conference Management Systems, Peer Review, Web Application, SvelteKit, Svelte, MongoDB

1 Introduction

The increasing scale and complexity of academic and scientific conferences require efficient digital tools for management and collaboration. As higher education institutions (such as universities) and their research centers organize multiple events annually, there is a pressing need for an internal, cost-effective system capable of handling submission, review, and scheduling processes.

Currently, research units within academic institutions, depend in general on third-party platforms such as EasyChair, which, while feature-rich, impose financial and functional limitations due to paid licensing tiers. This work proposes a proprietary, web-based conference management platform developed using modern JavaScript technologies (SvelteKit, TailwindCSS, and MongoDB) to replace such solutions and provide extensibility for future use within other organizational units.

The project's objectives were to: implement secure, authenticated access with permission levels; allow conference creation, paper submission, and review workflows; enable discussion between authors and reviewers; and support email templates and submission customization.

The research units previously relied on EasyChair for managing conference workflows. However, recurring costs and limited customization motivated the development of an in-house alternative.

The project was carried out within an internship in the third year of the BSc in Informatics Engineering. It provided a learning opportunity in modern full-stack web development, emphasizing usability, security, and maintainability.

The system was designed for use in organizational units of academic institutions, such as research and development groups, departments, and laboratories. The system should be reusable across multiple conferences, allow for the configuration of roles, permissions, and forms, and ensure scalability within the internal infrastructure of academic institutions.

2 State of the Art

The study of conference management systems reveals a mature yet fragmented ecosystem of software solutions developed over the past two decades. These systems typically address one or more of the following domains:

(1) submission and peer review management;
(2) event scheduling and program generation;
(3) participant communication and registration.

However, most tools either specialize in a single domain or present usability and cost limitations that restrict their broader adoption within academic institutions. This section reviews the main existing solutions and the relevant technologies used in modern web development that informed the design of the proposed system.

2.1 Existing Conference Management Systems

EasyChair is the most widely adopted online conference management platform in academia, supporting functionalities such as paper submission, blind and double-blind review, reviewer assignment, and decision management [2]. The system also offers event scheduling, statistics, and email communication. However, EasyChair's major drawbacks are its proprietary nature, high learning curve, and licensing costs for premium features such as advanced analytics and extended storage. Its interface, while functional, remains dated and unintuitive for first-time users. Additionally, conferences must be created by EasyChair administrators, limiting the autonomy of smaller organizations.

OpenConf represents one of the earliest open-source alternatives, providing extensive configurability and allowing full self-hosting [8]. It supports abstract and paper submission, customizable review forms, and schedule generation. Its *Community Edition* is free but requires manual installation and server maintenance, while *Professional Editions* introduce additional features and technical support. Despite its flexibility, OpenConf's user interface feels outdated and lacks modern UX standards, which limits adoption among non-technical organizers.

Oxford Abstracts focuses primarily on abstract submission and review workflows, simplifying the early stages of conference organization [9]. It features a highly intuitive interface, powerful filters, and reporting tools. However, the system does not support full paper submission or multi-track event scheduling, which reduces its suitability for large-scale academic events. Its reliance on paid licensing tiers also introduces similar cost barriers to EasyChair.

As shown in Table 1, a comparative analysis of these systems highlights their main features and limitations.

Table 1. Applications Feature Comparison

System / Feature	EasyChair	OpenConf	Oxford Abstracts	Proposed System
Paper submission	Yes	Yes	Abstract only	Yes
Peer review management	Yes	Yes	Yes	Yes
Schedule generation	Yes	Yes	Yes	Planned
Customization (forms, roles)	Limited	High (manual)	Moderate	High
Hosting	Proprietary	Self-hosted	Proprietary	Self-hosted
Cost	Paid tiers	Free / Paid	Paid	Free (internal)
Ease of use	Medium–Low	Low	High	High (target)

From the analysis, it becomes evident that no single platform provides an open, modern, and cost-free solution that balances usability, configurability, and extensibility for institutional needs. This gap motivated the creation of a customizable, open internal platform leveraging modern frameworks to combine the flexibility of OpenConf with the usability of Oxford Abstracts.

2.2 Web Technologies in Modern Conference Systems

Beyond functional comparison, the technological foundations of existing tools vary significantly. Traditional platforms were typically implemented using PHP and relational databases such as MySQL. Although reliable, these architectures lack the scalability and responsiveness demanded by today's web applications.

Modern systems, by contrast, are increasingly adopting JavaScript-based full-stack frameworks, leveraging the benefits of reactive interfaces, real-time updates, and serverless deployment [10, 12]. Three key technologies stand out in this context.

The evolution of web frameworks has been marked by the dominance of React, Angular, and Vue [10]. However, Svelte introduces a paradigm shift: instead of running the framework in the browser, Svelte compiles components at build time, generating minimal JavaScript bundles and delivering exceptional runtime performance [5].

SvelteKit, its companion meta-framework, integrates server-side rendering (SSR), routing, and API endpoints in a single unified environment [6]. This drastically simplifies full-stack development and avoids the complexity of maintaining separate frontend and backend applications.

For this project, SvelteKit's tight coupling between UI, logic, and routing proved ideal for prototyping, even though it imposed architectural constraints that limited horizontal scalability.

The choice of Prisma ORM with MongoDB Atlas provided a flexible and scalable backend. Prisma abstracts database interactions using a strongly typed schema and simplifies migration management.

Unlike SQL-based systems, MongoDB's document model enables flexible data storage - particularly suitable for heterogeneous entities such as conferences, papers, and reviews. Studies such as Chauhan [1] highlight MongoDB's efficiency in handling semi-structured research data, a core requirement for this system.

Tailwind CSS and DaisyUI were selected for the front-end styling, instead of traditional frameworks such as Bootstrap [13], to allow a utility-first approach, allowing developers to construct complex responsive layouts without maintaining large CSS files [3]. DaisyUI builds upon Tailwind to provide accessible, themeable UI components, accelerating development and ensuring consistent visual identity.

2.3 Architectural Trends

Modern conference management systems are increasingly adopting modular and layered architectures, such as the Onion or Hexagonal patterns [7, 11]. These models emphasize separation of concerns, testability, and independence from frameworks.

While SvelteKit's integrated architecture limits strict adherence to these patterns, this project applies the same principles conceptually: data flow moves from domain entities to application services, then to adapters (UI and persistence). This conceptual alignment ensures that, despite technical limitations, the system remains maintainable and logically organized.

From the literature and technological review, the following gaps were identified:

- Cost and licensing – Existing systems often require subscriptions for essential features.
- Limited customization – Most platforms offer fixed workflows not adaptable to specific conference types.
- Usability vs. flexibility trade-off – Tools with richer features (e.g., EasyChair) tend to be harder to use.
- Lack of open modern alternatives – Few systems exploit current-generation web frameworks such as SvelteKit or offer seamless integration of frontend and backend logic.

This context justified the design of a new, open, modular system that combines usability, configurability, and institutional autonomy, developed as a case study for academic institutions's research environment.

3 Analysis & Design

This chapter presents the main functional and non-functional requirements of the system.

Functional requirements included conference creation, role management, paper and review submission, email notifications, and discussion threads. Non-functional requirements emphasized usability, system security, performance — with page load times maintained under 2 seconds — and operational reliability.

The system follows a layered architecture inspired by the Onion Model, using SvelteKit to integrate front-end and back-end logic within a unified framework. From

a technological perspective, the frontend was built with SvelteKit, supported by TailwindCSS and DaisyUI for component styling. The backend logic was implemented inside SvelteKit using REST-like endpoints, with Prisma ORM managing persistence on a MongoDB Atlas cloud database. Authentication was ensured through Lucia-Auth, using hashed credentials and role-based permissions. Email notifications were implemented using NodeMailer, supported by dynamic template variables.

The main system actors are the Conference Manager, Author, Reviewer, and Administrator. The core use cases include conference creation and configuration, paper submission and peer review, discussion between authors and reviewers, and email invitation and template customization. Fig. 1 depicts the overall use case diagram.

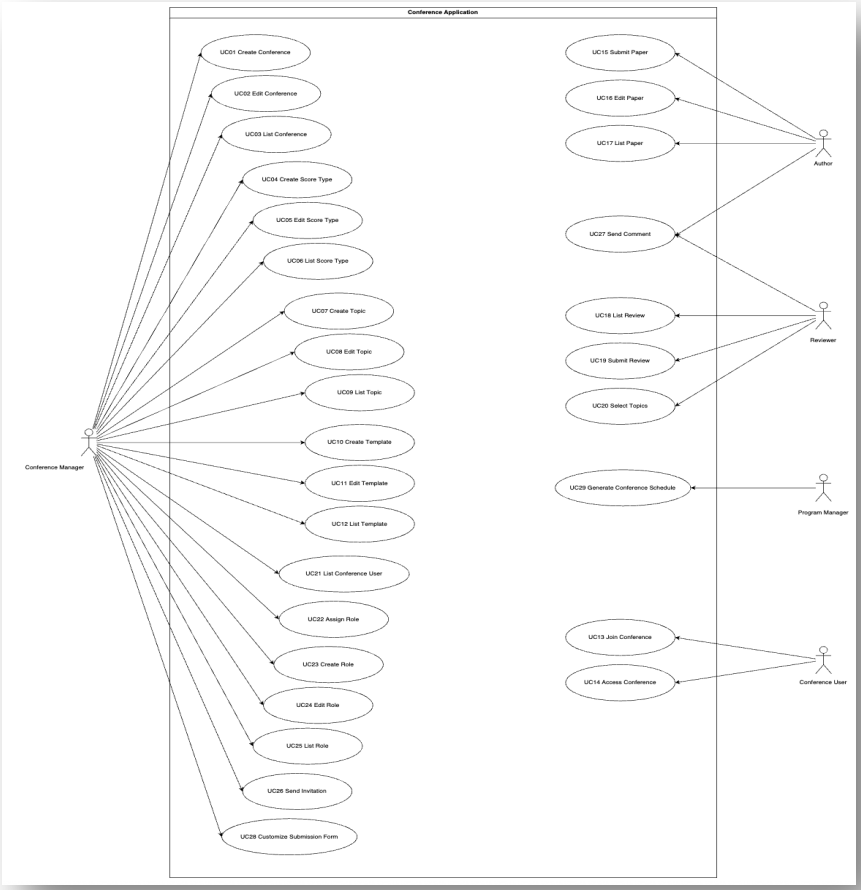

Fig. 1. Use Case Overview Diagram illustrating system actors and interactions.

The developed system is a web-based platform designed to manage the full lifecycle of academic conferences - from creation and submission to peer review and communication among participants.

It was built with a role-based access control model, ensuring that each user interacts only with the functionalities relevant to their role.

Access to the platform is handled through a secure login page, implemented using *Lucia-Auth* for credential verification.

Once authenticated, users are redirected to their personalized dashboard, which adapts according to their assigned role.

This approach enforces both security and functional separation between user types.

The Administrator oversees the entire system and manages global settings. Key functionalities include:

- Creating and managing user accounts and permissions;
- Monitoring all active conferences;
- Managing institutional parameters such as storage limits and access policies.

The administrator dashboard presents a dynamic table with filtering and pagination to simplify multi-conference management.

The Conference Manager is responsible for creating and managing a specific conference.

Their main features include:

- Conference creation (name, deadlines, topics, and configuration) (Fig. 2);
- Participant management (authors and reviewers);
- Reviewer assignment for submitted papers;
- Progress monitoring of the review process;
- Email communication through customizable templates (Fig. 3);
- Form customization for submissions.

The interface is divided into tabbed sections (General, Submissions, Reviews, Communications), allowing quick navigation across management areas.

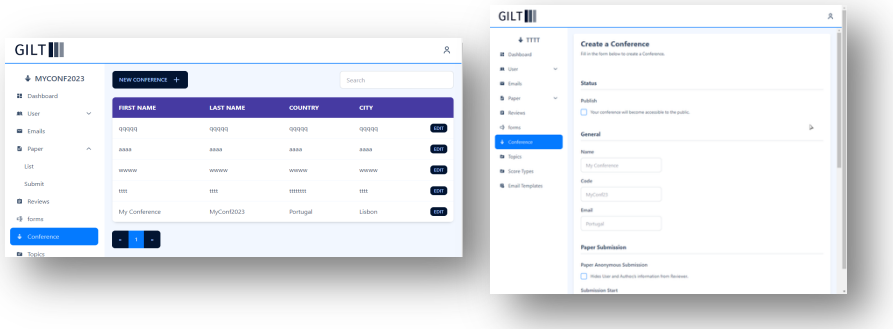

Fig. 2. Conference creation interfaces

Authors use the system to submit abstracts or full papers.

The submission process includes real-time validation using *Zod* and *Superforms*, ensuring correct and complete input.

Each submission is assigned a unique identifier and can be tracked through status indicators such as *Submitted*, *Under Review*, *Accepted*, or *Rejected*.

Authors can also:

- Edit or replace their submission before the deadline;

- View reviewer comments and feedback;
- Engage in direct discussion threads with reviewers.

The Reviewer interface provides access to assigned papers. Each paper entry includes:

- The uploaded file;
- A structured review form with scoring criteria (e.g., relevance, originality, clarity, recommendation);
- Comment fields for both the author and the conference manager.

After completing a review, the submission's status is automatically updated, and managers can view aggregated review data.

The system integrates an automated email notification module implemented with *NodeMailer*. Emails are triggered by key events such as:

- Submission confirmation;
- Reviewer assignment;
- Review completion and decision updates.

Conference managers can customize templates with dynamic placeholders (e.g., {author_name}, {paper_title}), allowing personalized automated communication.

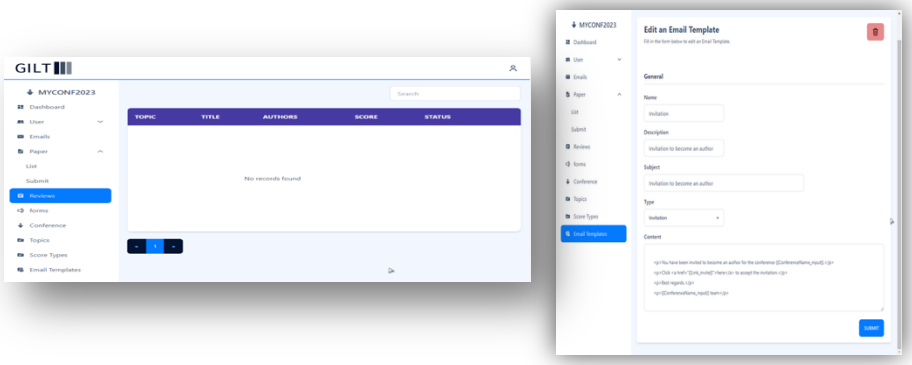

Fig. 3. Reviewer dashboard and email template interfaces

Thus, the application covers all essential stages of a scientific conference (creation, submission, review, and decision) within a unified, modern, and scalable platform. It provides academic institutions with a cost-free, customizable solution that balances usability, flexibility, and institutional autonomy.

4 Implementation

The project was developed using Visual Studio Code and Bitbucket for version control. The JavaScript runtime environment was Node.js [4], with pnpm managing dependencies. Testing was conducted via Playwright and MailCrab(for simulated email testing).

Originally, the system was designed as two separate applications (frontend and backend). Due to complexity in secure API integration, it was refactored into a single

SvelteKit monolithic application, which enabled faster development and easier integration at the cost of horizontal scalability.

This architectural choice is appropriate for prototyping and small-scale institutional use; however, in large-scale or production environments, it introduces limitations in terms of independent scaling and service decoupling. Alternative approaches could include a microservices-based architecture or a dedicated REST API backend separated from the frontend layer, as suggested in the future work section.

The frontend skeleton included: route-based navigation structure; layout inheritance (root and child layouts); TailwindCSS + DaisyUI integration; dynamic table components for pagination, filtering, and sorting.

Using Prisma ORM, models for users, conferences, papers, reviews, and templates were defined. Each entity had CRUD (Create, Read, Update, Delete) endpoints defined in +server.ts files.

Lucia-auth handled authentication, while roles were defined using binary strings for efficient permission storage.

The system implements several key functionalities to support conference management. Secure authentication and role-based access control have been fully implemented, ensuring that users can only access features appropriate to their assigned roles. Conference creation and editing is complete, allowing managers to define conference details, deadlines, and participant roles. The paper submission and editing module is fully operational, enabling authors to submit and modify their work before the deadlines. Similarly, the review submission and scoring functionality has been implemented, providing reviewers with structured forms to evaluate submissions and assign scores. An author–reviewer discussion feature is also available, facilitating direct communication and feedback exchange. In addition, the system supports email templates and invitations, allowing automated, customizable notifications to participants.

However, certain features remain unimplemented, including submission form customization, which would allow conference managers to fully tailor submission fields, and automatic schedule generation, which would automatically create event schedules based on submissions and reviews. These gaps represent potential areas for future development and enhancement.

The interface was built with SvelteKit, TailwindCSS, and DaisyUI, ensuring a responsive, consistent, and accessible design. The layout features a sidebar navigation menu and a main content area that adapts to different screen sizes. Reusable components (tables, forms, and modals) promote visual consistency and simplify maintenance.

5 Evaluation

Reusable form validation using Zod and Superforms streamlined user input validation. Server-side pagination was implemented manually, as no suitable Svelte-native library supported it.

Unit tests verified isolated functions, such as form validation and database queries. Although integration testing was challenging due to SvelteKit's tightly-coupled

architecture, End-to-End (E2E) testing using Playwright successfully validated core workflows including login, conference creation, and paper submission.

Due to the absence of an external test group, usability validation was performed internally. As a consequence, the results reflect a controlled and informal evaluation environment. Future iterations should involve real users — authors, reviewers, and conference managers — to enhance empirical validity.

Regarding performance, testing confirmed that page load times remained consistently below 2 seconds, and that database communication remained stable under normal operation conditions. However, no stress or large-scale concurrency testing was conducted, which restricts conclusions about scalability.

The project successfully demonstrated the feasibility of building a full-featured conference management system using SvelteKit. While the integrated architecture simplified deployment and accelerated development, it also introduced constraints in terms of modular scalability and distribution.

Lessons learned include:

- The importance of early architectural planning for scalability;
- The trade-off between framework simplicity and flexibility;
- The critical role of UI/UX considerations in user-centric systems.

The use of modern technologies - SvelteKit, TailwindCSS, Prisma, and MongoDBpr - oved efficient and educational, promoting deeper understanding of reactive programming and full-stack development.

6 Conclusion

The work presented in this paper achieved most of its original goals, producing a functional prototype of a Conference Management System tailored for academic institutions. It provides a viable internal alternative to EasyChair, offering cost savings and control over data.

In relation to the initial objectives, the system successfully implemented secure authentication, role-based access, conference creation, paper submission, review workflows, author–reviewer discussion, and email templates. However, two structurally significant modules — submission form customization and automated conference program generation — remain unimplemented. These features are considered essential for achieving full competitive parity with established platforms such as EasyChair and OpenConf and therefore represent high-priority areas for future development.

Future work should focus on: separating frontend and backend for horizontal scalability; implementing advanced authorization (JWT); completing missing modules and adding analytics dashboards; Enhancing accessibility and usability with user testing.

Despite limitations, the project delivered a meaningful, functional result and served as a strong foundation for continued development and institutional use.

References

1. A. Chauhan, “A review on various aspects of MongoDB,” *International Journal of Engineering Research & Technology (IJERT)*, 2019.
2. EasyChair, “About EasyChair.” Accessed: Dec. 3, 2025. [Online]. Available: https://easychair.org/
3. A. Fitzgerald, “Tailwind CSS: What it is, why use it,” *HubSpot*, 2022.
4. F. G. Ghansham Jadhav, “Role of Node.js in modern web application development,” *International Research Journal of Engineering and Technology (IRJET)*, 2020.
5. C. Humble, “All about Svelte, the much-loved, state-driven web framework,” *The New Stack*, 2021.
6. L. Lawson, “Rich Harris talks SvelteKit and what’s next for Svelte,” *The New Stack*, 2023.
7. R. Nunkesser, “Using Hexagonal Architecture for Mobile Applications,” pp. 113–120, 2022, doi: 10.5220/0011075100003266.
8. OpenConf, “OpenConf conference management system.” Accessed: Dec. 3, 2025. [Online]. Available: https://www.openconf.com/
9. Oxford Abstracts, “Abstract management and peer review software.” Accessed: Dec. 3, 2025. [Online]. Available: https://www.oxfordabstracts.com/
10. E. Saks, “JavaScript frameworks: Angular vs React vs Vue,” *Medium*, 2019.
11. J. Sidler, E. Braun, C. Schmitt, T. Schlachter, and V. Hagenmeyer, “Microservice-based Architecture for the Integration of Data Backends and Dashboard Applications,” *Karlsruhe Institute of Technology (KIT)*, 2024. [Online]. Available: https://publikationen.bibliothek.kit.edu/1000141315/136598801
12. S. Wildermuth, “Introducing Vite: A better Vue CLI?,” *CODE Magazine*, 2022.
13. P. P. Suraj Shahu Gaikwad, “A review paper on Bootstrap framework,” *IRE Journals*, 2019.