WEB TECHNOLOGIES AND SOFTWARE ENGINEERING

Travexa: A Web-Based Platform for Travel Assistance and Volunteer Coordination

Rajashri Kale, Shruti Kakde, Payal Fulari, Vaishnavi Damale

Department of Computer Engineering, Navsahyadri Group of Institutions, Pune, Maharashtra, India

Published November 2026

Abstract

Travexa is a web-based platform that brings travel assistance and volunteer coordination into one centralized digital system. It connects travelers, volunteers, hosts, and administrators around the management of travel-assistance requests. The platform is built with a Next.js frontend, a Node.js backend, and a PostgreSQL database, and volunteers are assigned to requests by an administrator or coordinator, which clarifies responsibilities and supports communication among stakeholders. This paper describes Travexa's architecture, user roles, workflow design, and implementation approach, and discusses its potential to improve the organization and accessibility of travel assistance services. Effectiveness, reliability, and scalability are addressed through a proposed evaluation framework; any quantitative performance claim would require validation through documented testing. Travexa aims to offer a structured and extensible foundation for digital travel assistance and community-based volunteer coordination.

Keywords: Access control; PostgreSQL; travel assistance; volunteer coordination; web-based platform; workflow management

1 Introduction

1.1 Background and Motivation

Travel supports education, work, healthcare, tourism, and social life, yet some people need help to complete a journey comfortably and safely. Elderly passengers, persons with disabilities, solo travelers, and those unfamiliar with a place may need support with navigation, luggage, finding facilities, or other specific requirements. Providing this support requires coordination between the person asking for help and the volunteers, hosts, or coordinators who can offer it.

When that coordination relies on telephone calls, messaging applications, and email, requests scatter across channels. Coordinators then find it hard to keep a shared view of request status, volunteer availability, and assignments, and they spend extra time collecting details and updating records. Travexa was motivated by the need for a single digital environment in which each request follows a structured workflow, with recorded data, role-appropriate interfaces, and traceable assignments.

1.2 Overview of Travexa

Travexa is a web-based platform for organizing travel-assistance requests and coordinating the people who deliver assistance. Its architecture separates the user interface, application services, and

persistent storage: the frontend is built with Next.js, the backend runs on Node.js, and PostgreSQL provides relational data storage. The platform supports four user roles:

- **Traveler:** Seeks travel-related assistance and uses the request-management functions.
- **Volunteer:** Provides assistance and accesses the functions associated with volunteer activities.
- **Host:** Supports the service ecosystem through the host-related functions available in Travexa.
- **Administrator:** Oversees the platform, including the coordination and assignment of volunteers to requests.

Volunteer assignment is carried out by an administrator or coordinator; it is not an autonomous matching process. The platform provides the environment in which requests and assignments are organized and tracked, but it does not independently verify the quality of assistance or guarantee that volunteers will be available.

1.3 Problem Statement

The problem addressed here is how to organize travel-assistance requests and coordinate service providers within one digital workflow. When records are fragmented, request details are spread across several channels, assignment decisions are hard to trace, and participants lack a consistent view of request status. These difficulties grow with the number of users, requests, and participating organizations. A web platform can help by providing structured records, defined responsibilities, and centralized assignment management, but an interface alone does not make a system reliable; functional correctness, access control, data consistency, and performance must also be examined.

1.4 Aim, Objectives, and Research Questions

The aim of this research is to design and study Travexa, a platform for travel assistance and volunteer coordination built on a Next.js frontend, a Node.js backend, and a PostgreSQL database. The objectives are:

1. To develop a centralized platform for managing travel-assistance requests.
2. To provide role-based interfaces for travelers, volunteers, hosts, and administrators.
3. To support administrator-led volunteer assignment through a structured request workflow.
4. To organize request and user information in a PostgreSQL database.
5. To evaluate functionality, usability, reliability, and performance through documented testing, and to identify limitations and future improvements.

The study addresses four research questions:

- **RQ1:** How can a web platform centralize travel-assistance requests and volunteer coordination?
- **RQ2:** How can role-based access support the distinct responsibilities of the four user roles?
- **RQ3:** How can administrator-led assignment improve the organization of assistance workflows?
- **RQ4:** Which evaluation methods suit the functionality, reliability, and performance of Travexa?

1.5 Scope and Contributions

The scope covers the design and implementation of the platform: frontend and backend architecture, relational data management, request handling, administrator-led assignment, and security considerations. Evaluation covers functional testing of implemented features and, where data are available, response performance, data consistency, and usability; quantitative findings are to be reported only

as the tests actually performed support them. Autonomous volunteer matching, predictive AI, live location tracking, and native mobile applications are not assumed unless verified in the implemented system, and are treated as possible future extensions.

The main contribution is the documentation and evaluation framework of a centralized, role-oriented approach to travel-assistance coordination. It comprises a multi-role architecture, an administrator-led assignment workflow, a database-backed organization of platform information, and a testing framework covering functionality, access control, data integrity, usability, and performance. The contribution is architectural and implementation-oriented; claims that Travexa reduces delays or outperforms existing platforms would need comparative or user-based evidence.

The remainder of the paper is organized as follows. Section 2 reviews related work and the research gap, Section 3 describes the architecture and design, Section 4 presents the methodology and implementation, Section 5 discusses results, Section 6 covers security, privacy, and ethics, Section 7 addresses limitations and future work, and Section 8 concludes.

2 Related work and research gap

2.1 Accessible Tourism Platforms and Stakeholder Coordination

Alves et al. [1] reviewed existing technological platforms for accessible tourism and proposed Access@tour by action, pointing to weaknesses in current platforms and a need for stronger collaboration and information sharing among stakeholders. Botti and Monda [2] examined how digital ecosystems can improve accessible tourism through information sharing and coordination among the actors who shape a trip. Travexa shares this stakeholder-oriented view, but its focus differs: Access@tour by action centers on accessibility information, whereas Travexa coordinates assistance requests and volunteer assignments.

These studies suggest that a technological solution should do more than publish information; it should structure how requirements are recorded and responsibilities communicated. Travexa applies this idea through separate roles for travelers, volunteers, hosts, and administrators and through administrator-led assignment. Whether this approach reduces delays or communication problems must be established empirically rather than assumed from the design.

2.2 Accessibility, Usability, and Evaluation

Eusébio et al. [3] evaluated the web accessibility of hotels, travel agencies, museums, and similar tourism organizations and found substantial barriers for people with disabilities, while Park et al. [4] reported recurring problems in the presentation and adaptability of tourism website content. Together these findings show that an online service is not automatically accessible. Teixeira et al. [5] assessed Access@tour by action through expert evaluation and end-user feedback, illustrating the value of examining both technical functionality and user experience.

For Travexa, inclusive design is therefore an evaluation requirement. Relevant criteria include keyboard navigation, readable text, sufficient color contrast, clear form labels, meaningful error messages, and compatibility with assistive technologies. Evaluation should also confirm that essential workflows can be completed, data are stored and retrieved correctly, role restrictions hold, and errors are handled properly. These are recommended criteria; accessibility compliance can only be established by testing the implemented interface against a recognized standard.

2.3 Research Gap

Earlier work shows the value of digital information, stakeholder collaboration, and inclusive design in accessible tourism [1, 6]; Nigg and Peters [6] specifically studied stakeholder collaboration in the development of ICTs for accessible tourism. Several questions remain open for a platform such as Travexa:

- **Request coordination:** whether a centralized workflow for receiving requests, assigning responsibilities, and monitoring progress works in practice.
- **Multi-role design:** how to organize distinct responsibilities clearly and restrict each role to its appropriate functions.
- **Volunteer assignment:** how effective administrator-led assignment is in terms of assignment time and request completion.
- **Technical reliability:** whether data handling, request processing, error management, and performance hold under varied usage.
- **Accessibility and usability:** a working interface does not guarantee an accessible one, so dedicated testing and user feedback are needed.

These are research opportunities, not proof that existing systems lack such capabilities. Travexa explores them through its implementation and systematic evaluation.

3 System architecture and design

3.1 Architectural Overview

Travexa follows a client–server architecture. Users work through a web browser; the frontend provides the interface, the backend processes authorized requests and applies business logic, and PostgreSQL stores the data. This yields three layers, shown in Fig. 1:

- **Presentation layer:** Next.js provides the user-facing pages and role-specific interfaces.
- **Application layer:** Node.js handles application logic, request processing, validation, and coordination between the frontend and database.
- **Data layer:** PostgreSQL stores application records according to the database schema.

Figure 1 shows the conceptual three-layer architecture of the Travexa platform.

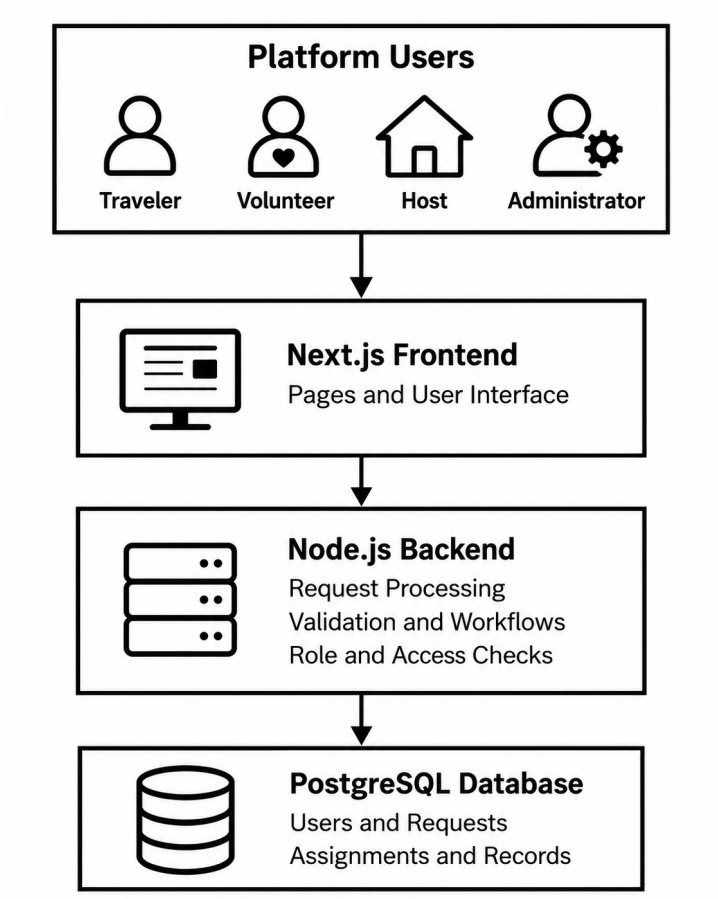


Figure 1. Conceptual three-layer architecture of the Travexa platform.

The diagram shows the proposed logical architecture and does not imply that every listed entity or backend function has been independently verified. The API framework, authentication mechanism, and schema should be documented from the deployed implementation.

3.2 User Roles and Responsibilities

1. **Traveler:** The traveler seeks assistance and, depending on the features implemented in the current version, submits or manages requests and reviews request information.
2. **Volunteer:** The volunteer assists with an assigned request; the interface is intended to provide the relevant assignment information and support the related responsibilities.
3. **Host:** The host contributes to the assistance process through the functions the platform assigns to this role; exact host permissions should be described from the implemented features.
4. **Administrator:** The administrator oversees platform operations, manages the assignment process, and performs the administrative operations the system permits.

Role separation matters because users should reach only the information and operations their role requires. These permissions should be enforced on the backend rather than relying on frontend navigation or hidden interface elements.

3.3 Volunteer Assignment Workflow

Travexa assigns volunteers through an administrator or coordinator rather than automatically, which keeps the allocation decision under administrative control. The intended workflow is:

1. A traveler submits an assistance request through the platform.
2. The request is recorded and made available to authorized personnel.
3. An administrator or coordinator reviews the request and relevant volunteer information.
4. The administrator assigns an appropriate volunteer.
5. The volunteer views the assignment information through the platform.
6. The request status is updated according to the supported workflow.

Request and assignment records are retained in the database where persistence is implemented. Evaluation should determine whether each stage operates correctly, whether unauthorized users can alter assignments, and whether the resulting records remain consistent. Practical effectiveness could be assessed through the assignment completion rate, the time taken to assign a volunteer, the request completion rate, and the number of unresolved requests.

3.4 Database Design and Design Considerations

PostgreSQL is the relational database in the Travexa stack; a relational model organizes platform information into connected entities while supporting constraints and consistent relationships. The proposed logical entities are users (identity, account information, role), assistance requests (details, requirements, status, timestamps), volunteer assignments (links between requests and volunteers), host information, and activity records of status changes and administrative actions where audit functionality exists. This is a proposed logical model rather than a verified description of the current tables, which should be documented with their keys and constraints. Integrity is especially important when several users interact with the same request: the system should reject invalid assignment references and validate status transitions, and transactions and constraints can help preserve consistency where implemented.

The design aims at maintainability and a clear separation of responsibilities. It calls for modular separation of interface, logic, and database operations; server-side permission checks for protected operations; input validation and valid request–assignment relationships; clear navigation and understandable error messages; scalability checks as users and concurrent operations grow; and protection of credentials and sensitive user information. These are design and evaluation requirements whose implementation must be demonstrated through code inspection and testing.

4 Methodology and implementation

4.1 Research Approach and Requirements

This study takes a system design and implementation approach, examining Travexa's architecture, roles, request workflow, assignment process, and technical evaluation. It proceeds in five stages: requirement analysis and identification of user roles; design of the architecture and data model; implementation of the frontend, backend, and database; functional and technical testing; and analysis of results and limitations.

Functional requirements cover the supported user operations, request handling, administrator-led assignment, and access to relevant information. Non-functional requirements include usability,

maintainability, data integrity, security, and acceptable response time. Requirements should be verified against the deployed application, since the presence of an interface element does not by itself show that an operation is fully implemented or persisted.

4.2 Technology Stack and Implementation

Travexa uses three confirmed technologies:

1. **Next.js frontend:** Builds the web interface and role-specific pages and communicates with backend services; its routing and rendering features help organize the frontend into maintainable components [7].
2. **Node.js backend:** Provides the server-side runtime that handles requests, validates submitted data, applies business rules, and communicates with the database.
3. **PostgreSQL database:** Offers persistent relational storage with constraints and transaction processing, which help maintain integrity when operations affect related records [8].

Choosing these technologies does not by itself establish performance, security, or scalability; these must be evaluated under defined conditions. In operation, the frontend sends user operations to the backend, which validates data, checks business rules, and executes authorized operations, reading or modifying PostgreSQL records where persistence is needed. Authoritative logic must stay on the server: an assignment action, for example, should be validated by the backend, and a user should not be able to reach another role's operations merely by altering a frontend request. Details such as the authentication mechanism, endpoints, and schema should be reported from the current code rather than inferred from the stack.

4.3 Testing Methodology

Testing checks whether Travexa performs its intended operations correctly. The proposed evaluation combines functional testing of expected outcomes, role-based access testing, data-integrity testing of stored information and relationships, usability assessment of task completion, and performance testing of response time under a documented workload. The following table defines the proposed test plan; these are test cases to execute, not reported results. The proposed test cases are listed in Table 1.

Table 1. Proposed test cases for Travexa.

Test ID	Test area	Test procedure	Expected outcome
T1	User access	Sign in using valid test credentials	Authorized access is granted
T2	Invalid access	Submit invalid credentials	Access is rejected
T3	Role permissions	Attempt an operation restricted to another role	Unauthorized operation is denied
T4	Request handling	Submit a valid assistance request	Request is processed according to the implemented workflow
T5	Input validation	Submit missing or invalid required fields	Appropriate validation response is returned
T6	Volunteer assignment	Assign a volunteer through the administrator workflow	Valid assignment is recorded if persistence is implemented

Test ID	Test area	Test procedure	Expected outcome
T7	Assignment integrity	Attempt an invalid assignment	Invalid assignment is rejected
T8	Data persistence	Retrieve a record after it has been saved	Stored data is returned correctly
T9	Status management	Update a request through a supported status transition	Status follows the defined workflow
T10	Performance	Measure response times under a documented workload	Measurements are recorded for analysis

Tests should be repeated where appropriate, with failures documented alongside the environment, input conditions, and expected and observed outcomes. Automated tests may be used where the implementation supports them [9].

4.4 Performance, Integrity, and Security Evaluation

Performance evaluation should examine response time, throughput, and error rate, reporting the average, median, and 95th-percentile response times. The average response time is calculated as

$$T_{\text{avg}} = \frac{1}{N}\sum_{i=1}^{N} T_i \tag{1}$$

where T_i is the response time of the i-th measured request and N is the number of measured requests. The 95th percentile captures slow requests that the average can hide; throughput is the number of successfully processed requests per unit time, and the error rate is the proportion of failed requests. For reproducibility, a report should state the hardware, software versions, deployment environment, test duration, request count, concurrency level, and measurement tool, and every figure must come from actual observations rather than estimates.

Integrity testing should check that invalid references are rejected, required fields are enforced, and updates leave the database consistent. PostgreSQL's transaction and concurrency control can help here [8], although correct behavior also depends on the application's transaction design and constraints. Security evaluation should cover authentication, authorization, input validation, protection of sensitive information, and confirmation that credentials do not appear in responses or logs; where sessions or tokens are used, their storage, expiry, and validation should be assessed.

5 Results and discussion

5.1 Evaluation Status

The evaluation concerns Travexa's architecture, role-based organization, request workflow, and administrator-led assignment. The design provides a centralized environment in which the Next.js frontend, Node.js backend, and PostgreSQL database separate user interaction, application processing, and data storage. Functionality described by the design must be distinguished from performance characteristics that need independent testing: claims about reliability, response time, data persistence, security, and usability should be reported as measured results only when supported by test records. Accordingly, this paper does not claim quantitative results such as the number of successful test cases, average response time, or uptime.

5.2 Functional Evaluation

The functional organization follows the responsibilities of each user role. The principal areas identified for evaluation are summarized in Table 2.

Table 2. Functional evaluation areas.

Functional area	Evaluation focus
User authentication	Examine whether valid and invalid login attempts are handled correctly.
Role-based access	Verify that users can access only the functions permitted for their roles.
Request management	Examine the handling and status management of assistance requests.
Volunteer assignment	Verify that administrators can assign volunteers through the supported workflow.
Input validation	Check how incomplete or invalid information is handled.
Data persistence	Verify that supported changes are stored and can be retrieved correctly.

The expected behavior of each function must be compared with its observed behavior during testing. Without documented execution and observed outcomes, the table is an evaluation framework and not evidence of successful testing.

5.3 Architecture, Data Integrity, and Access Control

Separating the frontend, backend, and database supports a modular organization: the frontend handles presentation and interaction, the backend validates and processes operations, and the database stores records. This gives a technical base for further evaluation, but performance, scalability, and reliability under real workloads remain to be measured. Data integrity and access control are central because the platform stores user information, requests, and assignments and serves users with different responsibilities. The implementation should be examined to confirm that valid operations are stored correctly, invalid ones are rejected, and each role can use only its permitted functions. The architecture and role definitions identify these requirements but do not prove they are met, so code inspection and the tests in Section IV are needed.

5.4 Discussion and Summary of Findings

Travexa offers a structured approach to organizing travel-assistance requests and volunteer coordination. Defined roles, a centralized request workflow, and administrator-led assignment are intended to give each participant clearer responsibilities, and the frontend–backend–database separation provides a basis for maintenance and future extension.

The discussion therefore rests on the architecture and intended workflows rather than on measured outcomes. A comparison with an existing coordination process, together with feedback from intended users, would be needed to show whether Travexa improves coordination, reduces delays, or enhances accessibility. The principal outcome of this paper is the architectural and workflow documentation of Travexa together with a framework for evaluating its functionality, access control, data integrity, usability, and performance. Quantitative results and claims of successful testing are not presented because they require documented test logs.

6 Security, privacy, and ethical considerations

6.1 Security Considerations

Because Travexa handles user and request information, security must be considered across the frontend, backend, and database layers. The system should verify identity before granting access to protected functions; restrict each user to the operations of their role; validate submitted input to reduce invalid data and common application attacks; use HTTPS for client–server traffic; avoid storing passwords as plain text, using an appropriate hashing mechanism if password-based authentication is implemented; protect database credentials and apply least-privilege database access; and keep error messages and logs free of sensitive implementation details and unnecessary personal information. These are recommended requirements; the actual controls must be confirmed through implementation review and testing.

6.2 Privacy and Ethical Considerations

Assistance requests may contain personal details, travel plans, contact information, or accessibility needs. Travexa should follow data-minimization principles, collecting only what the service requires, informing users how their data are used, stored, and shared, establishing retention and deletion procedures, and keeping request details from unrelated users. Administrative access should be limited to legitimate operational needs and reviewed periodically.

Ethically, the platform serves people who may need additional help, so it should obtain consent before collecting or sharing personal information, avoid discriminatory treatment in volunteer assignment, give clear information about request status and assignment decisions, offer accessible interfaces, provide a way to report inappropriate conduct or unresolved requests, and avoid claiming guaranteed safety or assistance unless operating procedures support it. In any user study, participants should be told the purpose of the evaluation and how their responses will be used, and their personal information should be protected throughout.

7 Limitations and future work

7.1 Limitations

1. **Limited independent evaluation:** Quantitative performance claims need documented test logs and reproducible measurements before they can be treated as verified.
2. **Dependence on administrator-led assignment:** Allocation depends on the administrator or coordinator, so efficiency may vary with request volume, volunteer availability, and administrative workload.
3. **Need for user-based assessment:** Technical testing cannot show whether the four user groups find the platform easy to use or whether it improves their experience.
4. **Accessibility and scalability:** Accessibility must be assessed against appropriate standards, and behavior under many users and concurrent requests requires load testing.
5. **Scope of operational claims:** Automatic volunteer matching, real-time GPS tracking, and native mobile applications are not established capabilities unless verified in the current implementation.

7.2 Future Work

- **Volunteer recommendation:** Suggesting volunteers by availability, skills, location, and request requirements, with transparent criteria and fairness evaluation.
- **Real-time notifications:** Timely updates on assignment decisions and status changes.
- **Accessibility improvements:** Evaluation against WCAG guidelines, keyboard-navigation and screen-reader testing, and usability studies with intended users.
- **Mobile and scalability work:** Improving the mobile-browser experience, assessing the need for a dedicated app, and load testing to guide database and infrastructure optimization.
- **Analytics and comparative study:** Aggregated metrics on assignment times and completion rates that respect privacy, and a controlled comparison with a manual coordination process.

These developments should be prioritized according to user needs, feasibility, privacy requirements, and evaluation evidence.

8 Conclusion

Travexa is a web-based platform that organizes travel assistance and volunteer coordination through a centralized system. Its confirmed technology stack is Next.js, Node.js, and PostgreSQL, and its user roles are traveler, volunteer, host, and administrator, with volunteers assigned by an administrator or coordinator rather than by an automated matching process. This paper has described the motivation, objectives, related literature, architecture, implementation approach, and evaluation framework, along with security, privacy, accessibility, and future development considerations.

The platform offers a foundation for studying how a multi-role web application can structure assistance requests and volunteer responsibilities. Conclusions about performance, reliability, usability, and real-world effectiveness, however, must rest on reproducible tests and user-based evidence. Future work should prioritize documented technical evaluation, accessibility validation, and assessment of the platform's usefulness to its stakeholders; its long-term value will depend on reliable workflows and the protection of user information.

Acknowledgment

The authors thank the Department of Computer Engineering, Navsahyadri Group of Institutions, Pune, Maharashtra, India, affiliated with Savitribai Phule Pune University, for the academic environment and support that made this work possible. They also thank the faculty members for their guidance and encouragement, and everyone who offered feedback during the design and development of Travexa.

References

[1] Alves J, Teixeira P, Eusébio C, Teixeira L. Benchmarking of Technological Platforms for Accessible Tourism: A Study Resulting in an Innovative Solution—Access@tour. *Applied Sciences*. 2022;12(8):3963. https://doi.org/10.3390/app12083963

[2] Botti G, Monda A. Finding a way towards high-quality, accessible tourism: the role of digital ecosystems. *The TQM Journal*. 2021;33(1):205–221. https://doi.org/10.1108/TQM-03-2020-0062

[3] Eusébio C, Teixeira L, Teixeira P, Caneiro MJ, Lemos D, Silveiro A. The State of Web Accessibility for tourists with disabilities: a comparative study between different tourism supply agents. *Disability and Rehabilitation: Assistive Technology*. 2023;18(1):17–29. https://doi.org/10.1080/17483107.2021.1941313

[4] Park K, Jeon H, Park SB. Disability e-inclusion for accessible tourism websites. *Current Issues in Tourism*. 2022;25(22):3571–3578. https://doi.org/10.1080/13683500.2022.2106195

[5] Teixeira P, Teixeira L, Eusébio C. Evaluating both usability and accessibility: the case of access@tour by action—a digital solution for accessible tourism. *Disability and Rehabilitation: Assistive Technology*. 2025;20(4):1159–1175. https://doi.org/10.1080/17483107.2024.2409810

[6] Nigg JJ, Peters M. The evolution of ICTs in accessible tourism: A stakeholder collaboration analysis. *Journal of Hospitality and Tourism Management*. 2022;52:287–294. https://doi.org/10.1016/j.jhtm.2022.07.007

[7] Vercel . *Next.js Documentation*. [Online]. Available: https://nextjs.org/docs. Accessed: Oct. 24. 2026.

[8] PostgreSQL Global Development Group . *PostgreSQL Documentation: Concurrency Control*. [Online]. Available: https://www.postgresql.org/docs/current/mvcc.html. Accessed: Oct. 24. 2026.

[9] OpenJS Foundation . *Node.js Documentation*. [Online]. Available: https://nodejs.org/docs/latest/api/. Accessed: Oct. 24. 2026.