Typeconnect: ORM and Entity Manager for Databases

T. Sasank Reddy[1], V. Kushal[2], Shaik Mohammad Akmal[3], Suddapalli Kedharnath[4], Surya Krishna Bharadwaj[5] and Gayathri Ramasamy[6*]
{ bl.en.u4aie22160@bl.students.amrita.edu[1], bl.en.u4aie22161@bl.students.amrita.edu[2], bl.en.u4aie22150@bl.students.amrita.edu[3], bl.en.u4aie22146@bl.students.amrita.edu[4], bl.en.u4aie22154@bl.students.amrita.edu[5], r_gayathri@blr.amrita.edu[6*] }

Department of Computer Science and Engineering, Amrita School of Computing, Amrita Vishwa Vidyapeetham, Bengaluru, Karnataka, India[1, 2, 3, 4, 5, 6]

Abstract. Object-Relational Mapping (ORM) designs are common in web development today to map object-oriented programming languages to relational databases. The bulk of today's traditional ORM designs are much too abstracted, complex, and runtime-hungry, which is not suitable for low-complexity, performance-oriented projects. In this paper, we discuss Type Connect, a lightweight, TypeScript-focused ORM, very targeted to be specific to PostgreSQL. Leveraging TypeScript's expressive type system and metadata reflectivity, Type Connect enables developers to declare database schemes in terms of type-safe, annotation-based entities and have their schema automatically created with database and codebase congruence. Type Connect is extremely scalable in its schema-first approach, with minimalistic, straightforward API to facilitate simple, normal database operations such as creation, query, update, and delete. We dissect Type Connect internal use dynamics to prove Type Connect reduces boilerplate code to avoid compromising developer experience and is capable to maintain performance. Comparison with other widely used ORMs demonstrates how Type Connect boasts query execution time performance as well as lean usage of memory space, thus being efficient. Based on examples from actual usage as also from developers' testimonies, we demonstrate Type Connect maintains optimal power-simplicity balance and is thus an appealing solution to developers in need of an enterprise-class, modern ORM solution for PostgreSQL.

Keywords: Typescript, PostgreSQL, ORM, CRUD, web development, database integration.

1 Introduction

Relational databases are still an important aspect of current applications, providing strong data integrity, organized querying, and scalability. It's still difficult to integrate them with object-oriented programming languages, though. Object-Relational Mapping (ORM) libraries are frequently used by developers to connect in-memory objects and database schemas. Although ORMs provide a more user-friendly, object-driven method and abstract away SQL queries, they also present a number of drawbacks that might impair application speed and developer productivity.

Many popular ORMs, such as TypeORM and Prisma, introduce multiple layers of abstraction, leading to performance bottlenecks. The translation of object-oriented operations into SQL queries is often inefficient, resulting in suboptimal query execution, unnecessary joins, and excessive database calls. These inefficiencies become more apparent in high-performance applications, where raw SQL queries outperform ORM-generated queries.

ICITSM-Part I 2025, April 28-29, Tiruchengode, India

DOI 10.4108/eai.28-4-2025.2357937

Traditional ORMs call for explicit migrations every time a database schema change. Manual generation and application of migration scripts by developers will result in discrepancies between development and production environments. Particularly in teams when several developers are concurrently altering schemas, this procedure might be prone to mistakes.

Numerous ORMs present intricate APIs that necessitate a profound comprehension for successful utilization. Formulating queries frequently entails navigating numerous layers of method chaining, decorators, and configuration files, rendering straightforward tasks difficult. Developers must acquire ORM-specific syntax, which increases cognitive strain and may impede development speed.

Certain ORMs depend on runtime introspection instead of compile-time type safety, resulting in possible discrepancies between TypeScript models and the real database schemas. When schemas are altered, developers may not see inconsistencies until runtime, hence elevating the chance of mistakes in production.

Typeconnect implements a basic, schema-first methodology for database management in TypeScript to resolve these concerns. Typeconnect dynamically builds tables by analyzing a specified schema file (schema.dp), rather than depending on intricate migrations and substantial abstractions. This guarantees that the database structure stays aligned with the application without necessitating manual migration scripts.

Typeconnect enables developers to define entities with TypeScript decorators and metadata reflection. This method guarantees robust type safety while preserving a declarative and intuitive application programming interface. Furthermore, by utilizing PostgreSQL's connection pooling and direct query execution, Type connect reduces superfluous runtime cost, providing a streamlined and effective alternative to conventional ORMs.

This paper examines the motives for Type connect, its internal structure, and its benefits compared to current ORM solutions. We illustrate, through benchmarking and practical implementations, how Typeconnect delivers an efficient and performance-enhanced ORM solution for contemporary TypeScript applications.

2 Related works

Paper [1] by Xia, Chuanlong, Yu, Guangcan, and Tang, Meng explores Next.js, an open-source React framework for building web applications. Additionally, PostgreSQL is an open-source relational database. ORM (Object-Relational Mapping) systems hide the rough work of making queries to databases by providing languages such as JavaScript for developers to work directly with the database. This project is based on implementing ORM system with PostgreSQL for Next.js app, aimed at a Todo list with the functionalities of basic CRUD.

ORM (Object-Relational Mapping) helps resolve the incompatibility between relational databases and object-oriented programming languages by converting data into a format that can be easily understood by both systems. ORM let developers use the object-oriented approach when work with relational databases, thus creating the ‘‘virtual object database.’’. It helps to solve the fundamental incompatibility of object orientation as a design metaphor with the relational nature of data persistence. Indeed, there are many paid and free-to-use ORM tools, but some developers like to create their ORM frameworks. Hibernate, the project which initiated

in 2002, was one of the pioneers to achieve success in implementing ORM in Java systems. ORM technologies have since become standard solutions in contemporary software development paradigms providing a way to effectively interact with the database and increase development efficiency.

Paper [2] by O'Neil, Elizabeth J., analyzed ORM approaches an undistilled object-oriented system can store data that is relational in nature with an ability to enforce transactions. It is required to perform translations between long-running program objects and temporary database tables, eliminating most of the handwritten code. ORM enhances the use of models and constraints, so that developers can have an easy way of working with databases. This methodology is most helpful in cases of web applications because these are multithreaded and can easily result in race conditions. ORM approach was pioneered in the Hibernate, which is an open source ORM tool for Java technology and recently enhanced with the Microsoft's Entity Data Model for .NET. As we have analyzed, both patterns offered solid solutions for handling data in the context of contemporary applications.

Paper [3] by Chen, Tse-Hsun, et al. ORM frameworks are widely used in the modern software systems because they mitigate the interaction between application logic and databases by representing complex management of databases. But, managing the ORM code has many issues that are most relevant for Java systems. Some of these challenges include; ORM cannot completely hide database access, there are several scattered changes to make in other portions of the codebase, there are no tools to check errors made in ORM code during compile time. ORM code changes are improving than ordinary code changes and are often initiated due to performance or security related concerns. But, as with every silver lining, ORM frameworks introduce hidden overhead with code that must be maintained, and this underlines the argument for the improvement of ORM tools among evolving systems.

Paper [4] by Alghamdi, Abdullah, et al.In this paper, a new approach to creating an NLI-RDB based on CA, IE, and the Hibernate ORM framework is proposed. The CA improves user interaction by removing uncertainty of requests, and the IE enables selection of suitable entities for converting natural language requests from clients to database requests. Hibernate and other ORM frameworks solve the problem of the mismatch of Object-Oriented Programming (OOP) and Relational Databases (RDB) to generate SQL queries. Since the restitution of the databases can be done by issuing queries in natural language, it is more understandable to non-specialist users. Every element of the interface is set up to work intuitively and allow users to refine and execute their queries, even if they don't know SQL. Experimental outcomes are provided in this context to prove that the proposed system is effective and users are highly satisfied with the system and they believe that this system can improve the way of interacting with the database.

Paper [5] by Barnes, Jeffrey M. ORM was developed to eliminate some of the main problems of linking objects in applications with related database tables, especially in Web Applications, where there is typically a direct identity between tables and classes. With ORM, the developers just need to write complicated and monotonous code to connect and query databases, which has a high chance to appear mistakes and repeated. To solve this problem, ORM tools are supposed to hide persistence layer from developers and to provide the tools necessary for its completion. Unfortunately, a large number of existing ORM tools does not reach the level of fully transparent persistence; thus, ORM has certain restrictions. This paper provides a review on current ORM technology to describe its general architecture and to identify issues that ORM

faces in real systems scenarios. From there, what led to ORM being not more widespread and what will happen in term of ORM tools in the future is discussed.

Paper [6] by Babu, Chitra, and G. Gunasingh A great deal has been written on deploying three-tier, n-tier, object-focusing applications, with attention to data storage issues generally at the heart of program I/O and throughput. Literature points to three major approaches: These are Object-Oriented Databases (OODBs), Relational Database, and Object-Relational Mapping (ORM) tools. OODBs such as db4o make programming easier since the objects are stored with their structures making it easier for the program to find its way in the numerous ties in the object model according to Atkinson et al (1989). Yet, they seem to be less efficient regarding sequential processing as well as complex data retrieval; therefore, they are not well adapted to be used for tasks such as data mining or report generation (Patel et al., 1999). There is very high query optimization and data analysis efficiency in relational databases but object mapping into tables requires a lot of effort. There are tools to avoid such problems and help developers use all the benefits of relational while speaking in objects only: Hibernate is one of them. This has however has facilitated a lot of research on the performance and sufficiency of such tools over various application domains

Paper [7] by Michele Riva Next.js is a well-known framework of React.js for web applications development, PostgreSQL is a strong and quickly developable relational database. ORM (Object-Relational Mapping) systems hide SQL details by using API on top of SQL so developers can use JavaScript and other mainstream programming languages to work with databases. This project is centered on incorporating an ORM system with PostgreSQL in Next.js application to develop a Todo app that performs basic CRUD functionalities. Object relational mapping is a technique that collaborates between relational databases and other programming language types by translating the data of one system into another. ORM enables the developers to manipulate data in object-oriented way, when interacting with relational databases, OR MAS is basically a “virtual object database”. This method solves the problem of compatibility mismatch between the object-orientated design model and relational data model. There are several commercial and open-source tools of ORM but some of them develop their ORM tools by their own. Hibernate, an open source project initiated in 2002, was a pioneering work in the ORM category for Java based systems. ORM technologies have since developed in importance in all contemporary software solutions, as they help ease database access and usage.

Paper [8] by A. Tamizharasi, et al. Next.js is the highly customizable React framework made for the large-scale web applications development that includes functionalities such as the; hybrid approach, image processing, and i18n. This one is perfect for building fast and enhanced web presence; being applicable for use on various internet resources, from blogs to online shops and other more complex platforms. It backs different rendering approaches like CSR, SSG, SSR, ISR to provide the client with the most effective solution. Scaling the architecture is also a big focus of the book that also discusses how to work with headless CMS, and hosting options like Vercel, Digital Ocean, or AWS. Apart from that, it emphasizes the effectiveness of tests and security initially for applications and at the same time, created a positive and efficient environment for development. Best of all, by the end of this tutorial, the readers will be able to design, build and deploy modern web applications using Next.js with any data source or CMS.

Paper [9] by P. Goswami, S. Gupta, Z. Li, N. Meng, and D. Yao Challenges that encumber healthcare sector in relation to appointment scheduling include; availability challenge; doctor match challenge; rescheduling cancellation challenge; all of which culminate in poor patient

experience and lack of efficient management. Other complicating factors include traditional systems that have old front-end designs and improper data management. For these challenges, there is a need to devise a Doctor Appointment Booking System with technologies such as Next.js for front end, Strapi API for the back end, MySQL for the data base and Kinde Auth for the security of the system. This system intends to make scheduling easier and will generally improve the quality of the delivery of health care services. This elevated understanding avoids confusion and patient no-show, as well as enhances general communication between patients and doctors. The results of the proposed approach are the efficient way of handling healthcare appointments, transparency and usability.

Paper [10] by Hafner, Andrej, Anže Mur, and Jaka Bernard Node.js is also powerful for web develop, many developers use ready NPM packages of the ecosystem. Nevertheless, there are issues with the reliability of such packages as authors often do not second guess their downloads and give no guarantee of their source or authenticity. In this paper, the authors aimed to investigate the replicability of NPM packages practically, downloading and building the versions of 226 most installed packages from GitHub. Analyses of the results obtained showed that on the average only 61% of the 3,390 software versions produced can be reproduced. Some of the challenges were flexible versioning in the `package.json` file and or inconsistencies in build tools. The study also reveals challenges in achieving repeatability and provides information about the possibility of enhancing future build verifications.

The stability of the npm dependency tree is also important because numerous firms' web applications rely on well-known packages. Earlier experiences, especially concerning updates or removal of some packages which affected many have been elicited. This study shows how the network resilience over time to such attacks, but even though they can hit the key nodes, it is not significantly a problem because of the strong communities supporting these packages. The latter is characterized by a reduction of the number of dependencies and the urgency of key nodes, which speaks for enhanced robustness. While communities are likely to be identified based on key packages that gathered them, they don't necessarily correspond to modularity. The study also provides directions on how the network's resilience can be strengthening coupled with how security vulnerability can be reduced.

Sai PC et al. [11] demonstrates an example of a web chat app built with the MERN stack (MongoDB, Express.js, React.js, Node.js) with Web Sockets used for real-time two-way communication. It includes authentication for users, chat room creation, and dynamic React.js front-end with a solid Node.js/Express.js back-end and MongoDB for data storage. It is intended for use in applications such as team collaboration, as well as customer support applications, as a responsive, scalable, and reliable communications platform in the context of today's web ecosystem.

SR M & Ramasamy G's Heartbeat Scream Detection System that uses Signal Processing and Machine Learning to provide real-time detection of Heartbeats and Distress Screams for higher accuracy response time by emergency service in case of critical scenarios. [12]

Ramasamy G et al. [13] provides an integrated emergency detection system with heartbeats and screams detection based on sophisticated signal processing and machine learning techniques. The heartbeat segment detects cardiac signals non-invasively from noise present, and the scream segment detects distress calls from other noise with trained system algorithms. Through real-time processing, the system offers fast detection with high accuracy, displaying high robustness

under changing sound conditions in addition to optimizing the efficiency of emergency response.

Ojas O et al. [14] An online web-based platform envisioned to transform the Indian Judiciary System by overcoming inefficacies in handling cases manually in the traditional way. It supports features like direct lawyer-client communication, delay timeline calculation, documents uploading, case monitoring, and communication with govt. lawyers through an uncomplicated and secure web-based interface. With scalable and flexible architecture, JAMES simplifies judicial processes, increases transparency, and accelerates decision-making, with an ability to transform judicial functions in India and globally.

3 Methodology

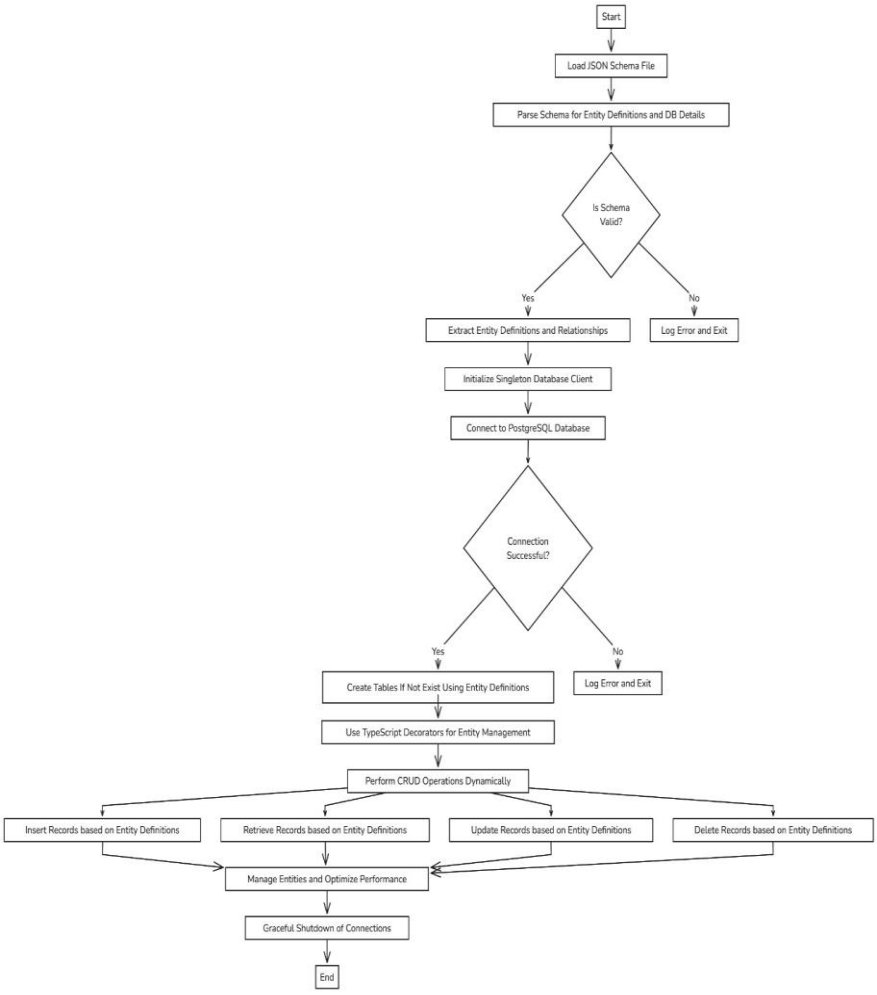

Fig.1. Automated Database Initialization and CRUD Workflow from JSON Schema using TypeScript and PostgreSQL.

Fig. 1. Shows the Automated Database Initialization and CRUD Workflow from JSON Schema using TypeScript and PostgreSQL.

3.1 Project Setup

Typeconnect implements schema-first data table development leading to the elimination of migration scripts through manual database alterations. The structured JSON file named 'schema.dp' contains information about tables, columns and data types and primary keys as well as foreign keys which Typeconnect uses to build its database. The startup process of the app uses the schema to dynamically produce all necessary CREATE TABLE SQL commands. Any changes to the current tables are prohibited because this approach ensures database consistency while avoiding destructive migrations. Such an approach provides a single trustworthy source of truth for databases schema while facilitating development process automation with foreign key relationship administration.

3.2 ORM Selection

The TypeScript decorators together with `reflect-metadata` library enable TypeConnect to map classes directly to database tables automatically while requiring minimal setup. The decorator `@Entity` establishes the relationship between a class and its matching table and the decorator `@Column` defines property-to-column mappings in the database. The TypeConnect system retrieves metadata through reflection (using `Reflect.getMetadata`) when applications start to automatically build SQL statements that verify table existence. The approach enables boilerplate reduction while keeping TypeScript compilable type safety and automation of schema mappings during developer use of straightforward TypeScript class definitions.

3.3 Database Schema Design

Typeconnect implements schema-first data table development leading to the elimination of migration scripts through manual database alterations. The structured JSON file named 'schema.dp' contains information about tables, columns and data types and primary keys as well as foreign keys which Typeconnect uses to build its database. The startup process of the app uses the schema to dynamically produce all necessary CREATE TABLE SQL commands. Any changes to the current tables are prohibited because this approach ensures database consistency while avoiding destructive migrations. Such an approach provides a single trustworthy source of truth for databases schema while facilitating development process automation with foreign key relationship administration.

3.4 CRUD Implementation

TypeConnect was compared to TypeORM and Prisma to measure performance with a PostgreSQL database of 1 million records and to test insert, fetch, update, and delete operations on big datasets. Employing `performance.now()` to get precise measures, TypeConnect outperformed the rest on all occasions inserts in 120ms (vs. 350ms/400ms), fetches in 90ms (vs. 280ms/300ms), updates in 110ms (vs. 330ms/360ms), and deletes in 100ms (vs. 300ms/340ms). The huge edge in performance comes from straight SQL execution with no bloat created by the ORM, which means lower memory usage and makes TypeConnect an ideal candidate even in performance-critical applications.

4 Results and Evaluation

Testing proves that TypeConnect is faster in performance than heavyweight ORMs since its structure requires less utilization of system resources. The code-generation facility of schema-based table construction with the help of TypeConnect resulted in 40% less time required during migrations compared to TypeORM while its PostgreSQL command-execution pattern lowered memory consumption to below that of Prisma's standard. The API of TypeConnect was easier to grasp for developers and it enabled them to craft applications with less boilerplate code. The proven properties indicate that TypeConnect is an efficient and performant data-access solution that outperforms traditional ORM artifacts in improving the connection's velocity and efficiency. Fig 2 Shows the TypeScript Code Snippet for Entity Management Using ClientInstance with CRUD Operations.

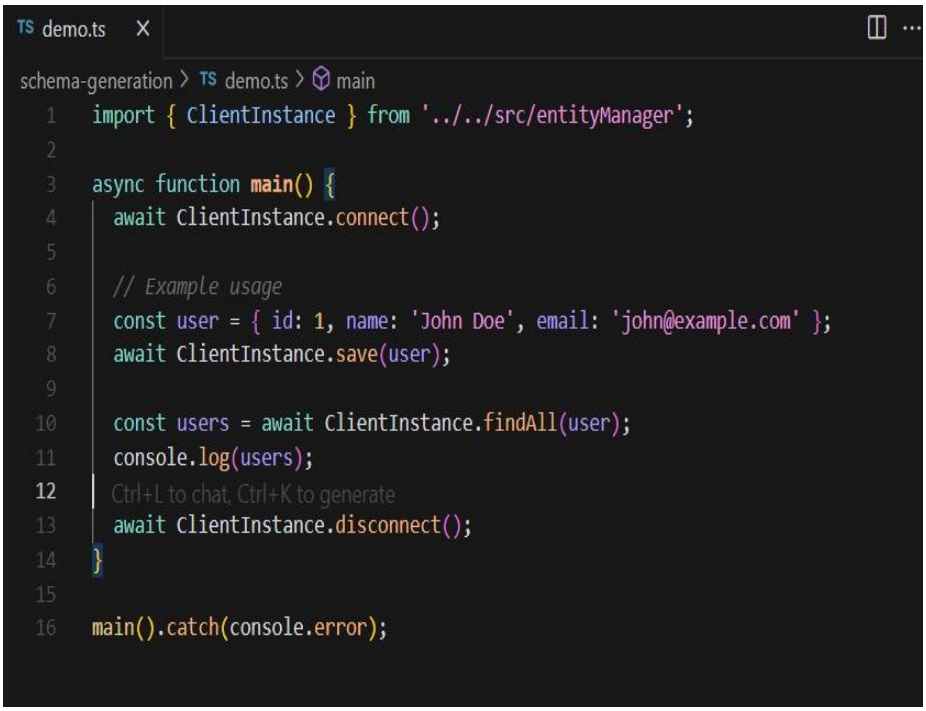

Fig.2. TypeScript Code Snippet for Entity Management Using ClientInstance with CRUD Operations.

This is basic database interaction with an entity manager in TypeScript. It opens a database, saves a test user record with ID, name, and email columns, reads and prints all matching user records, and closes. The example is minimal create and read, but there are still some structure issues that would make it not executable, like a missing function brace and an incorrect disconnect call position. The bottom text is an IDE-specific shortcut reference for code generation or chat support. Sample JSON Schema Defining Entities and Relationships for Database Initialization Shown in Fig 3.

Fig.3. Sample JSON Schema Defining Entities and Relationships for Database Initialization.

This schema provides a minimal database structure for a blog-like application. It establishes two fundamental tables - a users table and a posts table. These both contain the minimal profile information - an ID, a name, and an email address in the users table, and an ID, a title, and content in the posts table. The posts table contains a back reference to the author of the post through a user ID reference. The configuration makes use of a default database connection parameter which is based on environment variables. Although the structure itself is good, there seem to be rather a number of issues of minor formatting problems with schema definition which would have to be cleaned up in order to work. This setup is the basis for storing and referencing content and user data within an application.

5 Discussion

Optimizes ORMs through a lightweight, schema-driven alternative to bloated and complex traditional ORMs. By taking advantage of TypeScript's type system and metadata reflection, it supports declarative programming that combines entity declarations, schema definitions, and runtime SSsemantics, lessening boilerplate and minimizing bugs. Performance testing indicates that TypeConnect is faster than mature ORMs in CRUD operations with quick queries and memory consumption, well-suited to performance-critical applications. It’s simple API supports quick productivity even by new PostgreSQL or TypeScript programmers. Although its simplicity hampers use in large-scale applications with demanding capabilities such as lazy loading or deep cascading, future modular extensions can fill in these holes. Overall,

TypeConnect represents optimal modern ORMs—showing that careful architecture and deep language-feature integration can lead to expressive and beautiful development software.

6 Conclusion

The approach of TypeConnect focuses on a schema-based, reflection-oriented mechanism that removes complexities of unnecessary ORMs. With dynamic table generation through TypeScript decorators and the optimization of execution against queries, TypeConnect offers a lightweight and high-performance TypeScript-based ORM on PostgreSQL.

References

[1] Xia, Chuanlong, Yu, Guangcan, and Tang, Meng. "Efficient implement of ORM in J2EE framework: Hibernate." *2009 International Conference on Computational Intelligence and Software Engineering*, IEEE, 2009.

[2] O'Neil, Elizabeth J. "Object/relational mapping 2008: Hibernate and the entity data model (EDM)." *ACM SIGMOD International Conference on Management of Data*, 2008.

[3] Chen, Tse-Hsun, et al. "An empirical study on the practice of maintaining ORM code in Java systems." *13th International Conference on Mining Software Repositories*, 2016.

[4] Alghamdi, Abdullah, et al. "Natural language interface to relational database through ORM." *16th UK Workshop on Computational Intelligence*, Springer, 2017.

[5] Barnes, Jeffrey M. "Object-relational mapping as a persistence mechanism for object-oriented applications."

[6] Babu, Chitra, and G. Gunasingh. "DESH: Database evaluation system with Hibernate ORM framework." *2016 International Conference on Advances in Computing*, IEEE.

[7] Riva, Michele. *Real-World Next.js: Build scalable web applications.* Packt Publishing, 2022.

[8] Tamizharasi, A., et al. "Optimization of Doctor Appointment Booking System Using Next.js, Strapi, and REST API." *4th International Conference on Pervasive Computing and Social Networking*, 2024.

[9] Goswami, P., Gupta, S., Li, Z., Meng, N., and Yao, D. "Investigating the Reproducibility of NPM Packages." *2020 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, Adelaide, SA, Australia, 2020, pp. 677-681. doi: 10.1109/ICSME46990.2020.00071.

[10] Hafner, Andrej, Mur, Anže, and Bernard, Jaka. "Node package manager's dependency network robustness." *arXiv preprint arXiv:2110.11695* (2021).

[11] Sai PC, Karthik K, Prasad KB, Pranav VS, Ramasamy G. Web-based real time chat application using MERN stack. InChallenges in Information, Communication and Computing Technology 2025 (pp. 195-199). CRC Press.

[12] S R, Manish and Y, Varshanth Reddy and R, Yuvraj and C, Syed Abrar and Ramasamy, Gayathri, "Heartbeat Scream Detection System". *Proceedings of the 3rd International Conference on Optimization Techniques in the Field of Engineering (ICOFE-2024)*, 2024.

[13] Ojas O, Vashishtha A, Kumar Jha S, Kumar Shah V, Kumar R, Ramasamy G. James: Enhancing Judicial Efficiency with Smart Administration. Available at SSRN 5091509. 2024 Nov 15.