International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 09 Issue: 05 | May - 2025 SJIF Rating: 8.586 ISSN: 2582-3930

Generating Database Independent API Using AI

Prof.Atul Akotkar[1], Manya Dubey[2], Prajakta Udgirkar[3]

[1 2 3]*Computer Science and engineering & Nagarjuna Institute of engineering technology and management*

Abstract - In the rapidly changing software development environment of today, developing backend APIs that are not tied to particular database technologies is essential for ensuring flexibility, scalability, and compatibility across different platforms. This initiative suggests an AI-based approach to automatically create database-agnostic APIs, simplifying backend development and minimizing manual coding tasks. Utilizing machine learning and natural language processing methods, the system understands comprehensive user needs or schema specifications and produces fully operational RESTful APIs that facilitate CRUD operations, input validation, and error management. The created APIs work with various database systems (e.g., MySQL, PostgreSQL, MongoDB) by employing abstraction layers and ORM frameworks. This method enables developers to concentrate on business logic while speeding up development cycles, improving maintainability, and guaranteeing adaptability in various database settings. This solution is perfect for quick prototyping, corporate applications, and integration situations where backend adaptability is crucial.

***KeyWords*:**Artificial Intelligence (AI), RESTful APIs, Database Independence, Java Spring Boot, ORM (Object Relational Mapping), JPA (Java Persistence API*)*

1.INTRODUCTION

In today's fast-paced software development landscape, the demand for backend systems that are efficient, scalable, and maintainable has increased considerably. Developers frequently dedicate a significant amount of time to crafting boilerplate code for APIs that connect with databases. These APIs are generally closely integrated with particular database technologies, which complicates migration, maintenance, and support across different platforms. The goal of this project is to overcome this limitation by developing an intelligent system that automates the creation of database-independent APIs through Artificial Intelligence. This initiative was designed to reduce repetitive coding tasks on the backend and allow developers to concentrate more on essential functionality and business logic. Through the application of AI methods, the system analyzes input schemas and requirements to autonomously create APIs that are decoupled from any particular database. This leads to enhanced development velocity, uniformity in code quality, and increased adaptability in selecting or changing databases.

2. Methodology

This project showcases the creation of an AI-driven system that can autonomously produce database-agnostic RESTful APIs utilizing Spring Boot. The goal is to facilitate effortless API development across various SQL databases without having to code manually for each type of database. The backend structure adheres to a modular Spring Boot layered design that includes Controller, Service, and DAO tiers. Information is kept in SQL databases (MySQL/PostgreSQL), and Postman serves for API testing.

2.1 System Architecture

The architecture of the system is structured in layers: The Controller Layer processes incoming HTTP requests and aligns them with service methods. The Service Layer handles business logic and adjusts flexibly to various types of technology content. The DAO Layer handles database tasks through JPA, providing database abstraction. The Database Layer facilitates relational databases such as MySQL and PostgreSQL, which are abstracted through ORM.

2.2 Development Process

The procedure starts by analyzing SQL DDL input to create an abstract schema. From this, Spring Boot automatically creates entities, repositories, and controllers. APIs enable CRUD functionalities for content related to technology. Security is implemented through JWT authentication and role-based access control. The flexible design facilitates the straightforward integration of new databases or API functionalities.

2.3 Tools and Technologies

- Java 17 – Programming language for backend development
- Spring Boot 3.0.2 – Framework for APIs
- MySQL/PostgreSQL – relational databases using SQL.
- JPA (Hibernate) – Object-Relational Mapping for database relationships
- Postman – Testing APIs JWT – Security feature Git - Source code management

3. MODELING AND ANALYSIS

The system utilizes a multi-tiered Spring Boot architecture that includes Client, Controller, Service, DAO, and Database layers. This modular approach improves scalability, maintainability, and guarantees independence from databases.

3.1 Database Design

A normalized schema is employed to save technology-related articles featuring attributes like id, name, category, version,

description, posted_by, and posted_date. The database is compatible with MySQL and PostgreSQL via JPA and Hibernate.

3.2 Workflow

HTTP requests (GET, POST, PUT, DELETE) are dispatched from Postman to the Controller layer, which then sends them to the Service layer for handling business logic. The DAO layer manages database interactions, and replies are sent to the client, ensuring a distinct separation of responsibilities.

Figure 1: Workflow Digram.

3.3 Implementation

The backend APIs cleave to peaceful principles by exercising Spring Boot. ORM is managed through JPA/ Hibernate. JWT commemoratives cover the API endpoints, while Postman is extensively employed for testing and verification. Git oversees interpretation control and cooperation.

3.4 Requirements

Hardware: Intel i5 processor, 8GB RAM, SSD storage, and stable internet connection.**Software:** Spring Boot, SQL databases (MySQL/PostgreSQL), JPA/Hibernate, Postman, Git, and IDEs like IntelliJ IDEA or Eclipse.

4. RESULTS AND DISCUSSION

4.1 Result

The initiative effectively created a strong web application for submitting, modifying, and accessing technology-oriented information utilizing Spring Boot and SQL databases. RESTful APIs were developed for effective CRUD functions, tested using Postman. The system's tiered architecture provided modularity and ease of maintenance, while JWT implementation safeguarded API access. The platform facilitates the sharing of dynamic content across various technology types and versions. The design additionally accommodates upcoming features such as user roles, ratings, and comments.

4.2 Discussion

The design showcased how a modular, API- driven frame improves backend performance and scalability. Spring Boot simplified HTTP request operation, whereas JPA with Hibernate eased royal ORM. Postman played a crucial part in validating endpoints. Main advantages correspond of a structuredmulti-layered armature, secure authentication using JWT, and a regularized database schema that allows for extensibility. Obstacles, including setting up Spring Security and managing custom JPA queries, were dived via iterative debugging. The system establishes a solid base for forthcoming advancements similar as frontend UI, stoner analytics, and pall deployment.

5. CONCLUSIONS

The project effectively showcases the design and execution of an AI-driven system for API generation that is independent of any database. By leveraging Spring Boot, JPA, and SQL databases, the system offers a scalable and effective approach for developing RESTful APIs with reduced manual effort. The layered structure—consisting of controller, service, DAO, and database layers—guarantees a distinct division of responsibilities and encourages maintainability and scalability. During the development phase, the project adhered to contemporary software engineering practices, such as secure authentication via JWT, API testing through Postman, and version management using Git. The system's capability to abstract database functions via JPA enhances its versatility and allows for easy adaptation to various relational databases without the need to rewrite business logic**.**

ACKNOWLEDGEMENT

I wish to express my gratitude to my mentors, colleagues, and the organization for their unwavering support and guidance during the course of this project's development

REFERENCES

1. 1 Cole, J. B., & Florez, J. C. (2023). Standards of Medical Care in Diabetes—2023. Diabetes Care, 46(1), 377–390.
2. Lee, S., & Park, J. (2022). Applying Machine Learning to Database Schema Understanding for Automated Application Generation. Data & Knowledge Engineering, 139, 101900.
3. Jones, A., & Chen, Y. (2022). Leveraging AI for Automated Software Code Generation. IEEE Transactions on Software Engineering, 48(7), 236–249.
4. Kumar, V., & Singh, P. (2021). Security Challenges in Automated API Generation and Deployment. International Journal of Information Security, 20(4), 567–581.
5. OpenAI. (2021). Codex: An AI System for Code Generation and Understanding. arXiv preprint arXiv:2107.03374.
6. Smith, J., & Lee, M. (2021). Automated REST API Generation Using Schema Definitions. Journal of Software Engineering, 15(3), 112–124.