Industrial Engineering Journal
ISSN: 0970-2555
Volume : 54, Issue 8, No.1, August : 2025

STUBLAB - SIMPLIFYING API PROTOTYPING THROUGH APPLICATION SETUP AND SWAGGER INTEGRATION

Dr. S. Sudhakar, Computer Science and Engineering, SSM Institute of Engineering and Technology, Dindigul, Tamilnadu, India. sudhakars106@gmail.com
Dr. M. Nanmalar, Computer Science and Engineering, SSM Institute of Engineering and Technology, Dindigul, Tamilnadu, India. nanmalarm@gmail.com
M. Karthekeyan, Computer Science and Engineering, SSM Institute of Engineering and Technology, Dindigul, Tamilnadu, India. karthekeyan.muruganantham@gmail.com
R. Devaki, Computer Science and Engineering, SSM Institute of Engineering and Technology, Dindigul, Tamilnadu, India. devakiravi226@gmail.com
S. Harini, Computer Science and Engineering, SSM Institute of Engineering and Technology, Dindigul, Tamilnadu, India. harinisundar81002@gmail.com

ABSTRACT

Modern software development often encounters challenges in maintaining seamless coordination between frontend and backend teams, especially during early development and testing phases. Traditional backend systems rely heavily on database integration, making rapid iteration, testing, and collaboration complex and time-consuming. This paper presents StubLab, a backend stubbing tool developed using Java Spring- Boot for the backend and React.js for the frontend, designed to provide configurable mock APIs without the need for a persistent database. The system allows developers to define reusable request-response models, simulate real API interactions, and streamline integration testing workflows. StubLab enables users to create multiple applications, and within each application, developers can define and manage a wide range of APIs. Each API can be customized with different request and response structures, supporting versatile testing scenarios and parallel development. To further improve developer experience and maintain clarity, StubLab includes built-in support for Swagger documentation, automatically generating comprehensive and interactive API specs for all created endpoints. This fosters better communication, enhances collaboration, and accelerates the overall development cycle.

Index Terms— StubLab, Mock Server, API Testing, Swagger Documentation

1. INTRODUCTION

Backend development forms the backbone of modern soft- ware systems, yet it is often encumbered by complex setup requirements, tight database dependencies, and prolonged testing cycles. These limitations are particularly evident during the early stages of application development, where backend services are either incomplete or unavailable for frontend integration and testing. This disconnect slows down overall development and hampers collaboration between teams. To address these critical challenges, this paper introduces StubLab, a powerful backend stubbing platform developed using Java Spring Boot, designed to revolutionize API development and testing workflows [1]. StubLab provides a structured, database-independent environment that enables developers to create mock APIs [11] with highly configurable request-response models. These models are reusable, allowing teams to define once and iterate efficiently across multiple endpoints.

One of StubLab's standout features is its ability to generate mock data [9] automatically using AI-powered algorithms based on the defined models [3]. This significantly reduces manual effort, ensures data variety, and supports more robust testing. The platform allows users to create multiple applications [4] and, under each application, define and manage any number of APIs, making it highly scalable and adaptable to both small and large development projects.

To further enhance productivity, StubLab includes built-in Swagger documentation [6] [7] [8] [11] support, ensuring that all created APIs are instantly documented in a developer- friendly format.

This makes collaboration and onboarding easier for teams and helps maintain clarity across the software lifecycle.

StubLab is built with an API-first philosophy, empowering frontend and integration teams to move forward confidently without waiting on live backend systems. The platform is designed for ease of use, scalability, and extensibility, enabling quick prototyping, seamless team collaboration, and efficient integration testing.

By combining intuitive design, structured model-driven development [3], and AI-generated mock data [9], StubLab offers a transformative solution for backend stubbing. It aims to become a vital tool for developers, testers, and product teams by streamlining development workflows, reducing backend bottlenecks, and enabling rapid, reliable API simulation [2] [10]. This project envisions a shift toward smarter backend development, where teams can test faster, build better, and deliver more confidently—even without a live backend.

2. EXISTING SYSTEM

In today's development workflows, frontend and backend teams often find themselves stuck waiting on each other, which slows down the entire process. Front-end developers, in particular, face roadblocks when back-end APIs [2] [10] are not ready or are still being updated. In such cases, they're left with no choice but to work with static mock data or quick-fix stubs that lack flexibility, realism, and long-term usability. While tools like Postman, Mocky.io, and Mock Server exist to create mocks, they're usually limited to basic setups. These tools rely on manual configuration, offer static outputs, and don't support reusable models, making it harder to maintain consistency across environments. They also fall short when it comes to testing the full range of HTTP status codes, which is crucial for handling both expected and unexpected responses in real-world scenarios. Another major drawback is the absence of built-in Swagger documentation [7]. Teams often end up maintaining separate API docs manually, which leads to version mismatches and outdated references. On top of that, these systems don't use AI to generate mock data [9] dynamically, leaving developers to repeatedly create sample data by hand, which is both time- consuming and error-prone. The lack of contract-first development and automated [1] schema validation further complicates things. This results in frequent integration issues, duplicate efforts, and a slow de- bugging process. Plus, the documentation is usually scattered or missing entirely, making it hard for teams to stay on the same page.

In short, the current tools and methods fall short of what modern development teams need. There's no smart, centralized solution that supports parallel workflows, realistic response simulation, and auto-generated documentation—features that are essential for building fast, scalable, and reliable applications.

3. PROPOSED SYSTEM

To tackle the roadblocks in traditional API development, we bring in StubLab — a smart, model-first platform built to make API creation [2] and testing smooth, even when the backend isn't ready. StubLab changes the game by offering a centralized way to simulate API behavior [10] through structured stubs, so frontend teams can move forward without being held back by incomplete services or unreliable mock data.

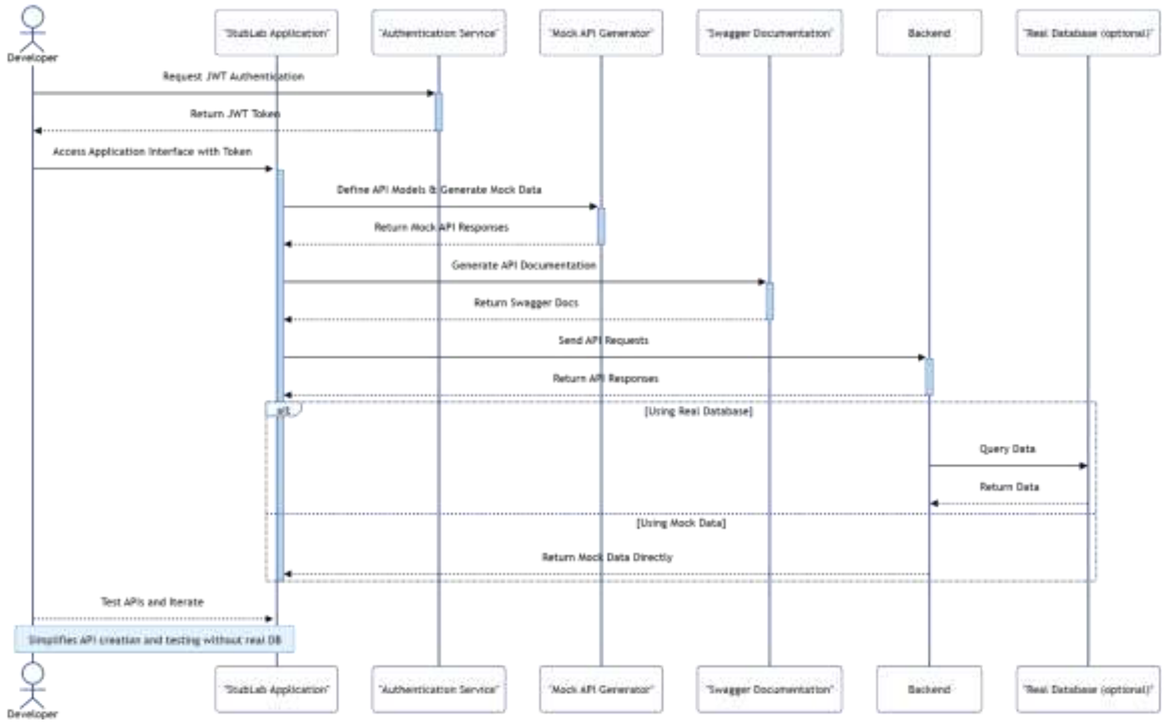
The main goal behind StubLab is to break the dependency chain between frontend and backend by letting developers define reusable request-response models. These models act as live contracts, powering mock responses that mimic real backend behavior. This brings consistency across environments, reduces back-and-forth debugging, and eliminates the guesswork during integration.

StubLab also comes with built-in validation tools that automatically check the integrity of API definitions. This ensures developers stay aligned with defined contracts, avoiding mismatches and schema issues. Everything happens through a simple and intuitive web interface, where users can easily create applications, build APIs under each app, and manage data flows effortlessly.

Unlike basic mocking tools, StubLab is made for collaboration and scale. Teams can share models, reuse API definitions, and maintain a single source of truth across projects [3]. This improves teamwork across frontend, backend, and QA units while cutting down redundancy and miscommunication. The platform runs on a strong Spring Boot backend, managing models, user authentication, and API logic efficiently. On the frontend, a React-based interface provides a responsive, clean experience that shortens the learning curve and boosts productivity.

To solve the documentation hassle, StubLab auto-generates live Swagger docs every time a model is created or updated [12]. This keeps API references current and accessible, so developers never have to maintain manual documentation again.

4. USERFLOW



This sequence diagram illustrates the workflow of API development and testing using StubLab, a mock API platform that simplifies the process by eliminating real database dependencies. The process begins with the developer requesting authentication from the Authentication Service, which returns a JWT token. The developer then accesses the StubLab Application with the token to define API models and generate mock data. This request is processed by the Mock API Generator, which returns mock responses. Simultaneously, the application triggers the Swagger Documentation module to auto-generate API docs, enhancing usability and clarity. Once setup is complete, the developer can test the APIs. The sequence includes an alternative flow (denoted by ”alt”) in one case, API requests are sent to the real backend, which queries an optional real database and returns actual data. In the other case, the system uses StubLab’s mock data, bypassing backend and database interactions entirely. This alternate path ensures rapid prototyping and testing. The final step reiterates the developer’s ability to iterate quickly using StubLab, ultimately streamlining API creation and testing without relying on a live database.

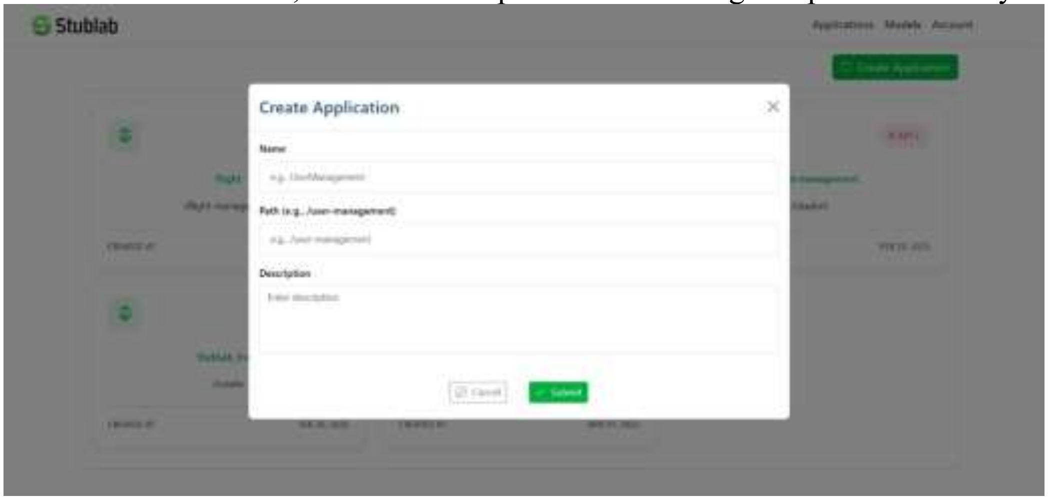
5. MODULE DESCRIPTION

A) Overview

This module enables users to create and manage applications that organize mock APIs under specific projects. Each application supports dynamic API creation with customizable paths, methods, and reusable models. Developers can simulate realistic API behavior without backend dependencies using structured stubs. StubLab also auto-generates Swagger documentation [8] for every application, ensuring real-time API visibility.

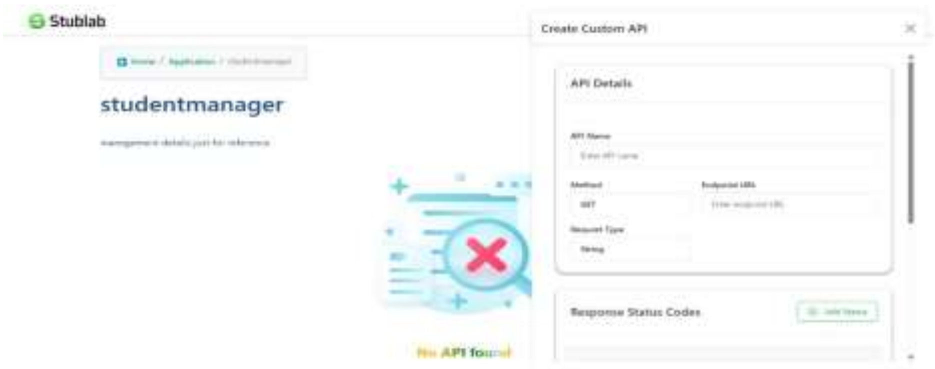
B) Application Creation

The Application Creation Page acts as the basis for efficiently managing and organizing mock APIs [9]. It allows users to create distinct applications, each representing a standalone project or service. These applications serve as logical containers where developers can group related APIs, models, and configurations under a single umbrella. This structured approach simplifies collaboration across teams and ensures a clean, scalable workspace for simulating complex backend systems.



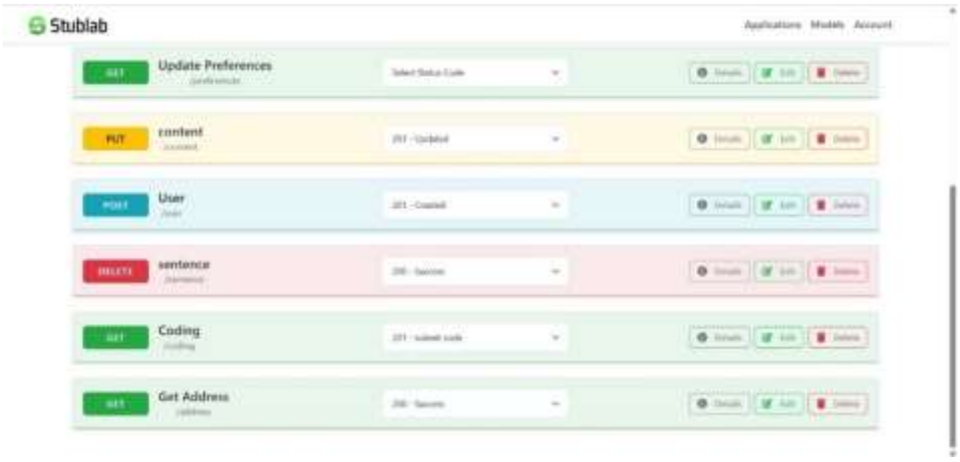
C) Add API Form

Within each application, the Add API Form [10] [2] plays a crucial role in defining mock endpoints. It enables users to input essential details such as the API path, HTTP method, and status codes and associate reusable request-response models to simulate realistic backend interactions. Developers can dynamically craft different scenarios like success responses, validation errors, or edge cases by configuring headers, query parameters, and payload structures. This modular API creation process helps teams test integrations thoroughly and develop frontend features without depending on live backend services.



D) Built-in Swagger

To complement the API stubbing functionality, StubLab offers built-in Swagger Documentation [6] that is automatically generated and always up to date. As soon as APIs are defined within an application, a live Swagger UI is made available, displaying the list of endpoints, supported operations, request formats, and example responses. Developers can even test these endpoints in real-time directly from the documentation interface [7]. This eliminates the need for manual documentation and ensures seamless communication between frontend and backend teams, enhancing productivity and reducing integration errors.



E) Frontend Module

The frontend is built using React to offer a seamless and user-friendly interface for interacting with all modules. Key technologies include: React Hook Form for building dynamic API input forms

• Axios for handling API communication
• React Router for navigation between application, API, and docs pages
• Embedded Swagger UI for rendering live API documentation

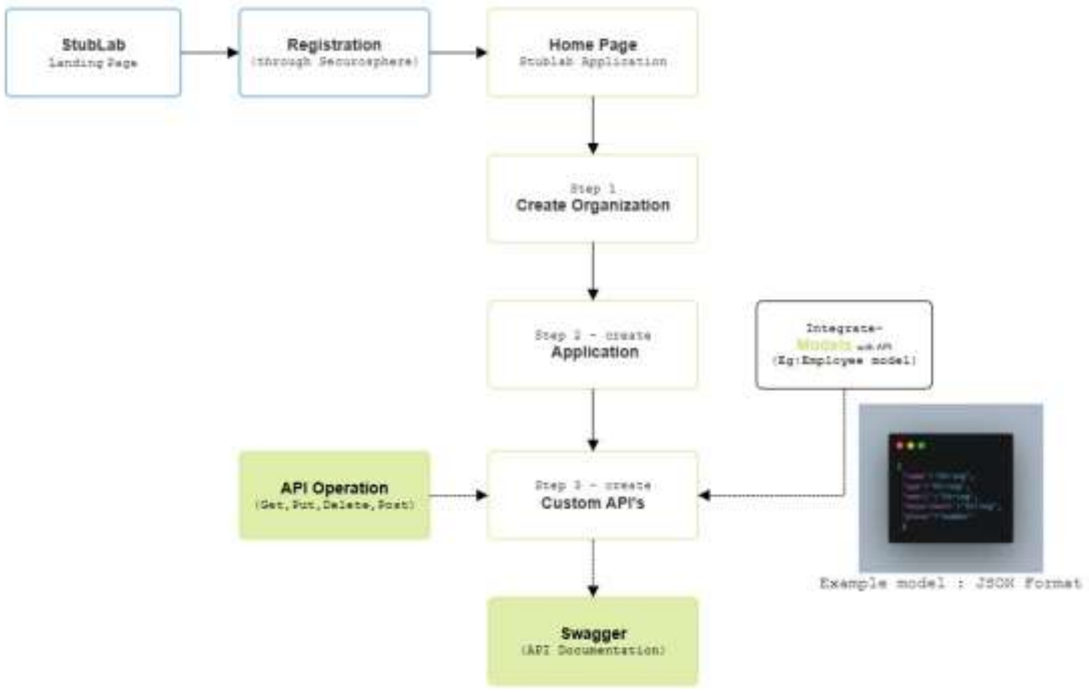
The frontend enables users to create applications, define APIs under them, and view real-time documentation, making backend simulation efficient and developer-friendly.

F) Backend Module

The backend of StubLab is developed using Spring Boot, exposing RESTful endpoints for managing:

• Application creation and listing
• API stub creation with request-response mapping
• Swagger documentation generation for each application

The APIs are stateless and designed for flexibility, enabling dynamic mock response generation using reusable models. Each application acts as a container for multiple APIs, and the backend ensures seamless interaction between modules, including Swagger auto-generation based on defined models.



6. TECHNOLOGY STACK

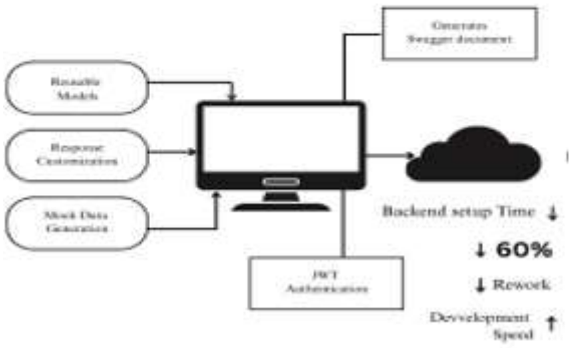
StubLab is built on a modern, scalable tech stack to streamline development, testing, deployment, and monitoring. The frontend uses React.js, with React Hook Form and Yup for dynamic form handling and validation, React Router for navigation, and React Toastify for real-time feedback. The backend, powered by Spring Boot, provides RESTful APIs for application creation, stub generation, and automated Swagger documentation. MongoDB stores dynamic models, API definitions, and app data efficiently.

Git and GitHub manage version control and collaboration, triggering Jenkins CI/CD pipelines. Docker ensures consistent environments, with Portainer simplifying container management. SonarQube maintains code quality, and Grafana delivers real-time system monitoring via custom dashboards. Figma supports collaborative UI/UX design.
For behavior analytics, Hotjar and PostHog track sessions, generate heatmaps, and analyze user flows for deep usability insights. StubLab's frontend is hosted on Vercel for fast, global deployment, while the backend runs on AWS EC2 with Docker containers and Nginx as a reverse proxy for enhanced security and scalability.

This integrated stack enables rapid development, robust testing, continuous delivery, and real-time feedback to drive StubLab's continuous improvement.

7. RESULT AND DISCUSSION

StubLab is a supercharged mock API server that removes backend and database dependencies, allowing teams to develop, test, and prototype APIs instantly. It enables fast API mocking with reusable models, real-time response customization, built-in Swagger docs, JWT authentication, AI-powered mock data generation, and 100% customizable status codes. StubLab cuts backend setup time by up to 60%, reduces rework by 30%, and boosts development speed across teams, all without needing a backend or database.



8. CONCLUSION

StubLab revolutionizes the backend development process by providing a model-driven, database-free API stubbing platform that accelerates frontend-backend collaboration. With dynamic model generation, inbuilt swagger documentation, and scalable deployment, StubLab ensures faster development cycles, im- proved integration testing, and enhanced team productivity. It lays a strong foundation for building modern, efficient, and agile software systems.

9. REFERENCES

[1] Abhinav Sharma, M Revathi, et al. Automated api testing. In 2018 3rd International Conference on Inventive Computation Technologies (ICICT), pages 788–791. IEEE, 2018.
[2] Alberto Martin-Lopez. Ai-driven web api testing. In Proceedings of the ACM/IEEE 42nd international conference on software engineering: companion proceedings, pages 202–205, 2020.
[3] Lauren Murphy, Mary Beth Kery, Oluwatosin Alliyu, Andrew Macvean, and Brad A Myers. Api designers in the field: Design practices and challenges for creating usable apis. In 2018 IEEE

Symposium on Visual Languages and Human-Centric Computing (VL/HCC), pages 249–258. IEEE, 2018.
[4] Ashwin Menkudle, Sheetal Sonawane, and Arvind Jagtap. Extracting application model from restful web services for client stub generation. Int. J. Comput. Technol. Appl. (IJCTA), 5(1):226–232, 2014.
[5] Omar S Go´mez, Rau´l H Rosero, and Karen Corte´s-Verd´ın. Crudyleaf: a dsl for generating spring boot rest apis from entity crud operations. Cybernetics and Information Technologies, 20(3):3–14, 2020.
[6] Je´ssica Soares Dos Santos, Leonardo Guerreiro Azevedo, Elton FS Soares, Raphael Melo Thiago, and Viviane Torres da Silva. Analysis of tools for rest contract specification in swagger/openapi. In ICEIS (2), pages 201–208, 2020.
[7] Saritha Kondapally. Transforming healthcare api development with generative ai: A dynamic and efficient swagger framework. Example Journal of Emerging Tech.
[8] Sandra Casas, Diana Cruz, Graciela Vidal, and Marcela Constanzo. Uses and applications of the openapi/swagger specification: a systematic mapping of the literature. In 2021 40th International Conference of the Chilean Computer Science Society (SCCC), pages 1–8. IEEE, 2021.
[9] Thilini Bhagya, Jens Dietrich, and Hans Guesgen. Generating mock skeletons for lightweight web-service testing. In 2019 26th Asia- Pacific Software Engineering Conference (APSEC), pages 181–188. IEEE, 2019.
[10] Venugopal Reddy Depa. The evolution of api management: Transform- ing modern integration landscapes. International Journal of Computer Engineering & Technology, 16(1):70–81, 2025.
[11] Jose´ Mat´ıas Rivero, Sebastian Heil, Julia´n Grigera, Martin Gaedke, and Gustavo Rossi. Mockapi: an agile approach supporting api-first web application development. In Web Engineering: 13th International Con- ference, ICWE 2013, Aalborg, Denmark, July 8-12, 2013. Proceedings 13, pages 7–21. Springer, 2013.
[12] Mahdi Jaberzadeh Ansari. An evaluation on automated technical documentation generator tools. The University of Calgary, 2022.