Bima Journal of Science and Technology, Vol. 8(3) Sept, 2024 ISSN: 2536-6041

DOI: 10.56892/bima.v8i3A.810

Development of a New Minimalistic PHP Micro Framework: NOAPHP

Muhammed Kabir Ahmed*, Gurama Auwal Umar and Muhammad Dawaki

Department of Computer Science, Gombe State University, Gombe, Gombe State, Nigeria

Corresponding Author: mkahmed@gsu.edu.ng

ABSTRACT

Web application development projects demand efficient frameworks to address the challenges associated with complexity and accessibility. Noaphp, a lightweight PHP micro framework, was developed as a solution to simplify the development process, targeting small to intermediate web applications. This research outlines the key features and significance of Noaphp, including a secured database integration, a validation engine for server-side validation, and a simplified and secured authentication module. Emphasizing the importance of a structured and approachable framework, Noaphp adopts the Model-View-Controller (MVC) architecture. This methodology ensures a clean separation of concerns, enhancing modularity, maintainability, and scalability. Noaphp offers commitment to fostering collaboration and innovation within the developer community, positioning itself as a gateway for understanding framework fundamentals.

Keywords: PHP Framework, Noaphp, lightweight framework, web framework, micro framework, PHP.

INTRODUCTION

In the digital landscape of the 21st century, where technology drives innovation, web application development has emerged as a cornerstone of software engineering. One notable statistic is that PHP, a popular server-side scripting language, powers approximately 79% of all websites whose server-side language is known (W3Techs, 2023) . This widespread adoption of PHP highlights its significance in the web development ecosystem. Furthermore, a recent survey by (Stack, 2024) revealed that PHP is the 10th most commonly used programming language among professional developers.

Beginners and intermediate PHP developers who haven't learned a framework often find it challenging to use larger frameworks like Laravel and Symfony for building simple web applications, as they are suitable for building large-scale and enterprise solutions (Majida et al., 2019). This results in a steep learning curve as they strive to master a framework and develop secure, robust web solutions (Barzilai, 2022) . Understanding the code can also be difficult since they might not fully understand how the framework processes data, making the projects maintenance very hard for beginners, sometimes impossible (Klika, 2017) . These challenges highlight the need for a PHP micro-framework that acts as a gateway for beginner developer, allows them to leverage their core PHP knowledge to write secure, small to intermediate web applications (Vermeulen, 2023).

In response to these challenges and the growing need for accessible web development tools, the NoaPHP micro-framework is introduced here. This framework is designed to simplify the development process, reduce the learning curve, and allow developers to build on their knowledge of the core PHP programming language. By offering a minimalistic approach, NoaPHP makes it easier for developers to create secure, small to intermediate web applications, thereby making web development more accessible to a broader audience. The remaining sections of the paper present literature review, design methodology of the new proposed php framework and its architecture. Thereafter installation guide, database configuration, documentation and a conclusion of the article was discussed.

LITERATURE REVIEW: PHP FRAMEWORKS

The landscape of web development has been significantly shaped by various PHP frameworks, which provide developers with tools and structures to build robust applications efficiently. Among the most popular PHP frameworks are Laravel, Symfony, and CodeIgniter, each offering distinct features and benefits that cater to different development needs. The functionality of a web application can set it apart in a crowded market. Frameworks with efficiency and performance optimizations might provide them a competitive edge, drawing in more users and possibly increasing conversion rates. (Jones, 2022).

Previous research and industry best practices established criteria that aided in the selection of PHP frameworks for this review. Scalability, performance features, popularity, and documentation quality were among the important factors (Potencier, 2020). Similarly performance indicators that are often used in performance evaluation literature were chosen, including average reaction time, throughput, error rates, and scalability (Davis & Patterson, 2019).

Laravel is widely recognized for its elegant syntax and powerful features, making it a favorite among developers. It incorporates a range of functionalities such as Eloquent ORM, Blade templating engine, and built-in authentication, which streamline the development process (Khouiji et al., 2019). According to a survey by Stack Overflow (2024), Laravel is one of the most commonly used PHP frameworks, indicating its popularity and effectiveness in the developer community.

Symfony, on the other hand, is known for its flexibility and scalability, making it suitable for large-scale enterprise applications. It follows the Model-View-Controller (MVC) architecture and promotes best practices in web development. The framework's modular component system allows developers to use only the parts they need, enhancing performance and maintainability (Khouiji, Abir, & Kerkeb, 2019)

As seen in table 1 below, while larger frameworks offer undeniable advantages, they also present some common challenges.

Table 1: Challenges of Larger PHP Framework*s*

CHALLENGES	DETAILS
STEEPER LEARNING CURVE	Mastering a complex framework demands significant time and effort, posing a hurdle for novice developers
PERFORMANCE CONSIDERATIONS	Larger frameworks can sometimes introduce overhead, impacting website speed and efficiency
OVERKILL FOR SMALLER PROJECTS	Utilizing a feature-rich framework for a simple website might be inefficient and unnecessary
FINDING THE RIGHT TOOLS	With a bunch of components and modules, choosing the right ones for your project can be a complex task

Micro Frameworks

Micro-frameworks are a category of software development frameworks designed to provide the minimal number of libraries and functionalities required to build applications. Their lightweight nature ensures that developers can avoid the bloat and complexity often associated with larger frameworks. Typically, micro-frameworks offer routing and some simple tools for request handling, response generation, and middleware integration. They are well-suited for constructing RESTful APIs, microservices, and other networked services where performance and resource utilization are critical. Designed for simplicity and speed, micro-frameworks are highly extensible through plugins or middleware, making them ideal for small to medium-sized applications (Bondar, 2023).

DESIGN METHODOLOGY OF NOAPHP

In the development of Noaphp, we have chosen to utilize an **Iterative** Software Development Life Cycle (SDLC) model. This decision is rooted in several key considerations that align with the project's goals and requirements.

MVC Pattern in Noaphp

The Model-View-Controller (MVC) pattern is a design pattern commonly used in web development to organize code in a structured and modular way. Noaphp follows the MVC pattern to separate concerns, making the framework more maintainable and scalable. Figure 1 shows the architecture of Noaphp.

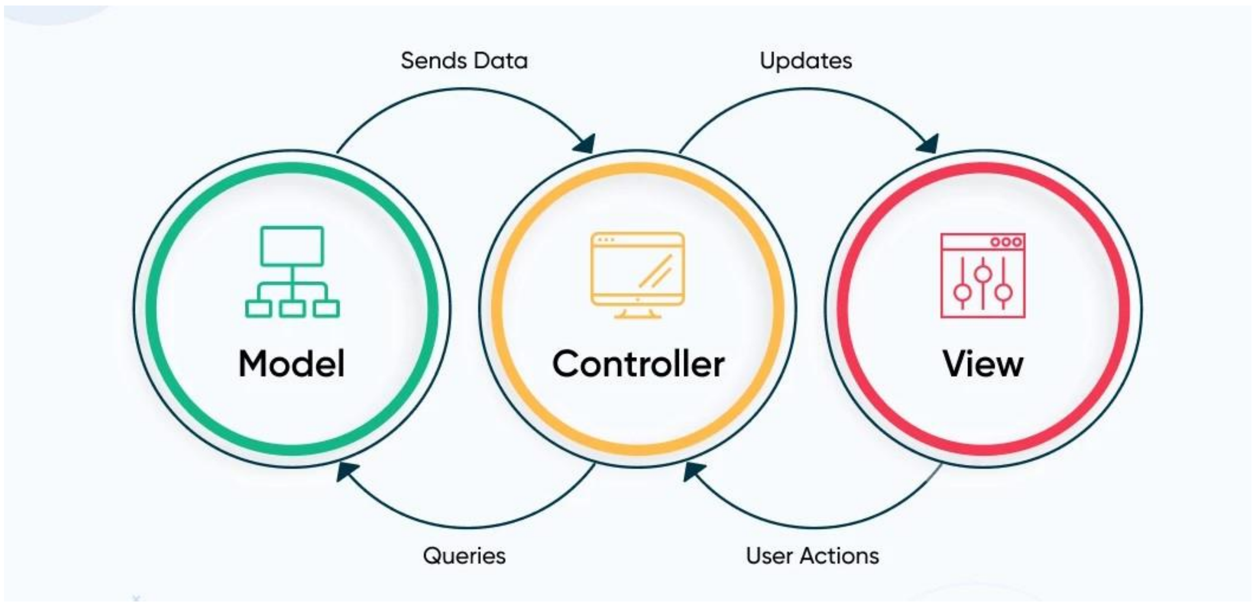

Figure 1: The architecture of Noaphp

Model: The Model represents the data and business logic of the application. In NoaPHP, the Model includes the data structures, database interactions, and any other logic related to handling data. It serves as the backbone of the application, managing the core functions that manipulate and retrieve data. The Model ensures that the application operates on a solid foundation of well-organized and efficiently processed information.

View: The View is responsible for presenting data to the user and handling user interfaces. In NoaPHP, the View includes templates that define the structure and layout of the pages that users see. It focuses on the visual aspect of the application, ensuring that data is displayed in a user-friendly and aesthetically pleasing manner. The View bridges the gap between the user and the underlying data, making it accessible and understandable.

Controller: The Controller manages the flow of the application, handling user input and updating the Model and View accordingly. In NoaPHP, the Controller receives user input, processes it, interacts with the Model to update data, and then passes the updated data to the View for rendering. It acts as an intermediary, coordinating the interactions between the Model and the View, ensuring that user actions lead to appropriate responses and updates within the application. Figure 2 present the stages of user request in Noaphp.

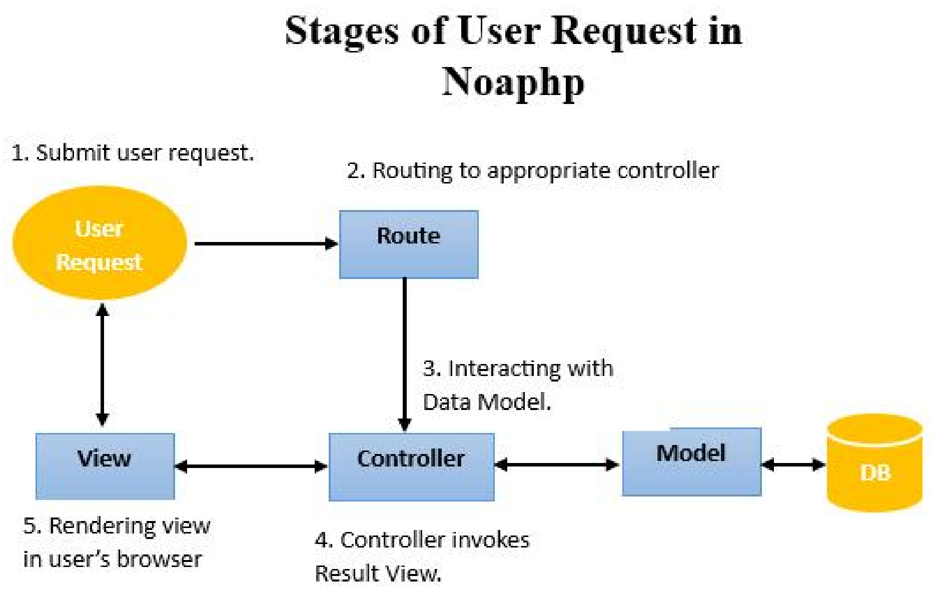
Stages of User Request in Noaphp
1. Submit user request.
2. Routing to appropriate controller
User Request
Route
3. Interacting with Data Model.
View
Controller
Model
DB
5. Rendering view in user's browser
4. Controller invokes Result View.

Figure 2: Noaphp user request handling

Stages of User Request in Noaphp:

1. Submitting User Request:

- The user interacts with the application by submitting a request, usually through a web browser. This request can be triggered by actions like clicking a button, submitting a form, or accessing a specific URL.

2. Routing to Appropriate Controller:

- The Noaphp framework has a routing system that examines the user's request and determines which controller should handle it.
- The routing mechanism maps URLs to specific controllers and actions, ensuring that the appropriate code is executed based on the user's request.

3. Interacting with Data Model:

Once the controller is identified, it interacts with the Model to perform any necessary data-related operations.
This may involve retrieving data from a database, updating records, or performing any business logic associated with the user's request.

4. Controller Invokes Result View:

After processing the user's request and interacting with the Model, the Controller invokes the appropriate View to generate the response.
The View is responsible for presenting the data in the desired format, typically HTML for web applications.

DOI: 10.56892/bima.v8i3A.810

5. Rendering View in User's Browser:

- The final step involves rendering the generated View and sending it back to the user's browser.
- The user's browser then displays the result, completing the cycle of user interaction.

By following the MVC pattern and these stages, Noaphp will ensures a separation of concerns, making the codebase more modular and easier to maintain. Each component (Model, View, Controller) has a distinct role, contributing to a well-organized and efficient web development framework. Also choosing this architecture will help developers to easily work on framework of their choice when going further, as MVC is the standard architecture used by majority of frameworks in PHP community.

Project Directory Structure

A well-defined project structure was established to organize the Noaphp framework efficiently. The structure is shown in figure 3.

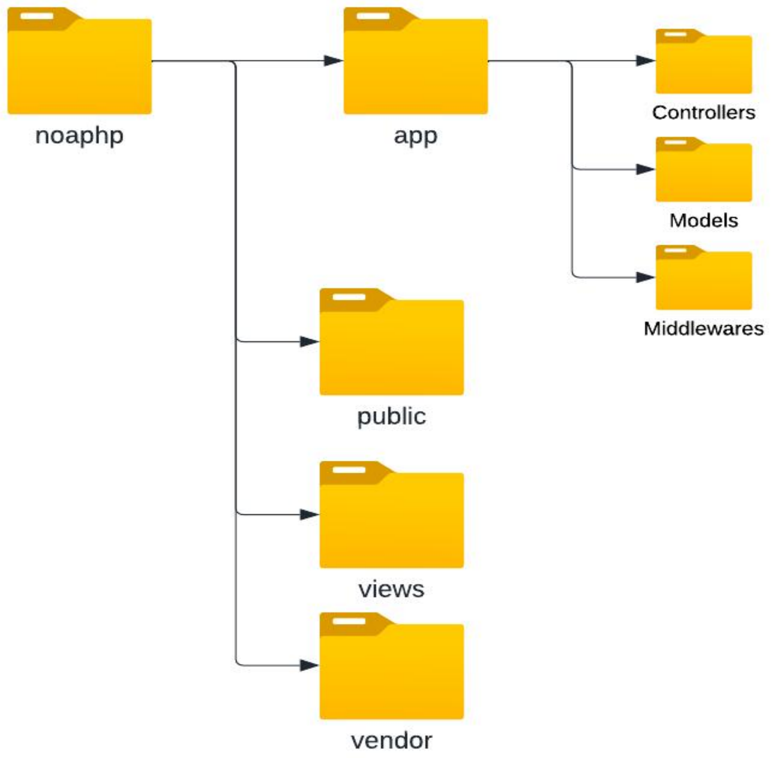

Figure 3: Noaphp project structure

Installation Steps

1. Clone the GitHub Repository:

Open your terminal and execute the following command to clone the Noaphp repository:

```bash
```

DOI: 10.56892/bima.v8i3A.810

```
        git clone https://github.com/noaphp/noaphp.git
```

2. Navigate to the Project Directory:

Move into the cloned project directory using the `cd` command:

```bash
        cd noaphp
```

3. Install Dependencies:

Run the following Composer command to install the project dependencies:

```bash
composer install
```

4. Serve the Application:

Use the following command to start the development server:

```bash
php -S localhost:8000
```

The application will be accessible at **`http://127.0.0.1:8000`** by default.

8. Access the Application:

Open your web browser and navigate to the provided address (e.g., `http://127.0.0.1:8000`). You should see the Noaphp framework running successfully.

Database Configuration:

If you need to set or modify the database configuration, update the relevant settings in the **app/settings.php** file.

Documentation

The documentation for Noaphp is a central resource accessible at www.noaphp.github.io. This comprehensive documentation serves as a vital asset throughout the development process, catering to various stakeholders and facilitating effective communication within the project.

Features of Noaphp micro framework

Noaphp, as a minimalistic PHP micro framework, significantly contributes to the knowledge and practice of web development. Here are some key aspects of its contribution:

1. **Reduced Learning Curve:** Noaphp minimizes the learning curve for new developers by focusing on simplicity and eliminating unnecessary complexities commonly found in larger PHP frameworks.

2. **Common Feature Implementation**: It contributes by implementing necessary features that are common to almost all web

applications, aiding developers in building robust and functional projects more efficiently.

3. **Simplified Structure**: Noaphp provides a simplified structure for developing simple to intermediate web applications. This contributes to creating more maintainable and understandable codebases.

4. **Scalability**: The framework is designed to scale and adapt to the growing needs of user requirements, ensuring that projects built with Noaphp can evolve with changing demands.

5. **Targeted Use:** By targeting small and intermediate web applications, Noaphp fills a niche in the PHP ecosystem, offering a specialized solution for projects of varying sizes.

6. **Accessibility**: It contributes to the accessibility of web development by offering a framework that is easy to learn and use, making it an ideal choice for developers entering the field.

7. **Gateway to Framework World:** Specifically tailored for PHP developers unfamiliar with frameworks, Noaphp provides a gentle introduction to the world of frameworks while maintaining an easy learning curve.

8. **Flexible Database Integration**: Easily integrate and interact with databases using our straightforward database layer. Noaphp provides flexibility while maintaining simplicity.

CONCLUSION

In conclusion, the NoaPHP micro-framework addresses the significant challenges faced by beginner and intermediate PHP developers when working with larger frameworks. By providing a simplified, minimalistic approach, NoaPHP lowers the learning curve and allows developers to leverage their core PHP knowledge to build secure and efficient web applications. The framework's clear separation of concerns into Model, View, and Controller components ensures that developers can manage data, user interfaces, and application flow with ease and clarity. Ultimately, NoaPHP democratizes web development, making it more accessible to a broader audience and empowering developers to create robust applications without the complexity and steep learning curve associated with larger frameworks. This framework represents a step forward in making web development more inclusive and manageable, fostering innovation and growth within the developer community.

REFERENCES

Barzilai, O. &. (2022). Using Web Frameworks in Server-Side Programming Courses. *Journal of Computer Information Systems*, 63(4), 866–876. https://doi.org/10.1080/08874417.2022.2111378.

Bondar, S. (2023). *Micro-frameworks - Streamlined Development for Web Services | Reintech media*. Retrieved from Reintech media: https://reintech.io/terms/category/understanding-micro-framework-in-software-development

Clutch. (2022). *The State of Web Application Development in 2023.* https://clutch.co/us/app-developers/government-industry.

Davis, J., & Patterson, D. (2019). CPU resource management for web servers. ACM Transactions on Computer Systems, 37(2), 1-30. https://doi.org/10.1145/3349870

Ferdinan, C. (2020). *Why Laravel Isn't For Begginers (And That's Okay).* https://medium.com/@arjunamrutiya/laravel-for-beginner-guide-mastering-the-fundamentals-d6d2b70dee67.

DOI: 10.56892/bima.v8i3A.810

Jones, E. (2022). Competitive advantage through web application performance. Business Technology Journal, 17(3), 45-60.

Khouiji, S., Abir, E., & Kerkeb, M. L. (2019). *A Comparative Study of Laravel and Symfony PHP Frameworks.* International Journal of Electrical and Computer Engineering (IJECE).

Klika, P. (2017). *Bachelor Thesis Developing web application using modern practices.* Prague.

Majida, L., Mohamed , K. L., Benmoussa, K., Khoulji , S., & El Yamami, A. (2019). A comparative study of laravel and symfony PHP frameworks. *International Journal of Electrical and Computer Engineering (IJECE)*, 710.

Potencier, F. (2020). Symfony 5: The fast track. Symfony SAS.

Stack, O. (2024). *Technology | 2024 Stack Overflow Developer Survey.* https://survey.stackoverflow.co/2024/technology#most-popular-technologies-language-prof.

Vermeulen, G. (2023). *Laravel vs Vanilla PHP: The Ultimate Showdown - Which one is the best for your next project?* https://www.linkedin.com/pulse/laravel-vs-vanilla-php-ultimate-showdown-which-one-best-vermeulen.

W3Techs. (2023). Retrieved from Usage statistics of PHP for websites: https://w3techs.com/technologies/details/pl-php