IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

Smart Deployment Assistant Using Generative AI for Automated Dockerfile Generation

Vaibhavi Harishwar Patil, Nikita Ramesh Rathod, Rutuja Bhagwan Sawai, Nikita Haribhau Ghadge

B.E Student, Department of Computer Science & Engineering
ICEEM, Waluj, Chhatrapati Sambhajinagar (Aurangabad), Maharashtra, India.

Prof. V. N. Shinde (Guide)
Department of Computer Science & Engineering, ICEEM,
Chhatrapati Sambhajinagar (Aurangabad), Maharashtra, India.

Abstract: Modern software deployment can be challenging, especially for beginners, as tools like Docker still require manual Dockerfile creation, which is complex and error-prone. This research introduces a Smart Deployment Assistant that uses Generative AI to automatically generate Dockerfiles from inputs such as GitHub repositories, ZIP files, or images. Built with Streamlit and Python, the system analyzes project data and produces optimized configurations, significantly reducing time and improving accuracy compared to manual methods. Overall, it simplifies deployment and makes it more accessible, efficient, and user-friendly for developers.

Index Terms – Generative AI, Docker, Dockerfile Automation, DevOps, Software Deployment, Streamlit, Containerization, Artificial Intelligence, Automation Tools, Cloud Deployment

I. INTRODUCTION

In recent years, software deployment has become more efficient with the use of automation and modern tools like Docker. However, writing Dockerfiles manually is still complex and error-prone, especially for beginners, which makes deployment difficult. To address this challenge, the project proposes a Smart Deployment Assistant that leverages Generative AI to automatically create Dockerfiles, simplifying and speeding up the deployment process. The system accepts inputs such as GitHub repositories, ZIP files, or images, processes them, and produces a suitable Dockerfile based on the selected project type“ The application is developed using Python for backend processing, with Streamlit providing an interactive and user-friendly interface. It provides features like project type selection, AI enable/disable options, and easy output visualization.

The main goal of this project is to simplify deployment, reduce errors, and make it accessible for students and beginner developers. Overall, it demonstrates how AI can improve software development by automating complex tasks and increasing productivity.

II. LITERATURE REVIEW

Modern software development increasingly relies on containerization, automation, and AI to improve deployment efficiency. Docker enables consistent application deployment but requires manual Dockerfile creation, which is complex for beginners. Tools like Jenkins and Kubernetes support automation but are difficult to configure. Existing solutions provide limited automation through static templates, which still require manual adjustments. With the rise of Artificial Intelligence, especially Generative AI, deployment tasks can be automated more intelligently by analyzing inputs and generating configurations. However, current systems lack integration between AI and deployment tools. The proposed Smart Deployment

Assistant addresses this gap by using AI to automatically generate Dockerfiles from inputs like GitHub, ZIP, or images, making deployment faster, easier, and more efficient.

III. RESEARCH METHODOLOGY

The methodology of the proposed Smart Deployment Assistant focuses on automating Dockerfile generation using Generative AI techniques. The system is structured so that each stage processes user input and converts it into a valid deployment configuration This workflow is designed to maintain simplicity, accuracy, and efficiency at every stage of the process. The system operates in a step-by-step manner, beginning with user authentication and continuing through to the final generation of the Dockerfile and deployment output.

3.1 User Authentication

The process starts with a login system that validates user credentials. This step ensures that only authenticated users can access the application. After successful authentication, the user is directed to the system's main dashboard.

3.2 Input Selection

After logging in, the user selects the type of input they want to provide. The system supports multiple input methods, including:

- GitHub repository URL
- ZIP file upload
- Image upload

It enables users to submit project data in different formats according to their preference.

3.3 Project Type Selection

The user selects the type of project they are working on, such as:

- Python
- Node.js
- Java

This selection helps the system understand the application's structure and requirements, which is essential for generating a suitable Dockerfile.

3.4 Data Processing

Once the input is provided, the system processes the data. Depending on the input type:

- GitHub URL → Project details are analyzed
- ZIP file → Files are extracted and examined
- Image → Basic assumptions are made based on the input context

The system identifies key components such as dependencies, runtime environment, and execution commands.

3.4 Output Generation

After processing, the generated Dockerfile is displayed on the screen. The user can:

- View the Dockerfile
- Download the file
- Proceed with deployment

The system also validates the output to ensure it contains the necessary Docker instructions.

3.4 Deployment Simulation

The system offers a simulation of the deployment process, allowing users to see how the generated Dockerfile would function in real applications.

IV. SYSTEM ARCHITECTURE

The system starts by verifying the user through the login module. After successful authentication, the user is taken to the Streamlit interface, where they can provide input in different forms, such as a GitHub repository, a ZIP file, or an image. The submitted input is processed to extract important project details. These details are then analyzed by the AI engine to generate an appropriate Dockerfile. The generated Dockerfile is shown to the user, who can either download it or proceed with a simulated deployment.

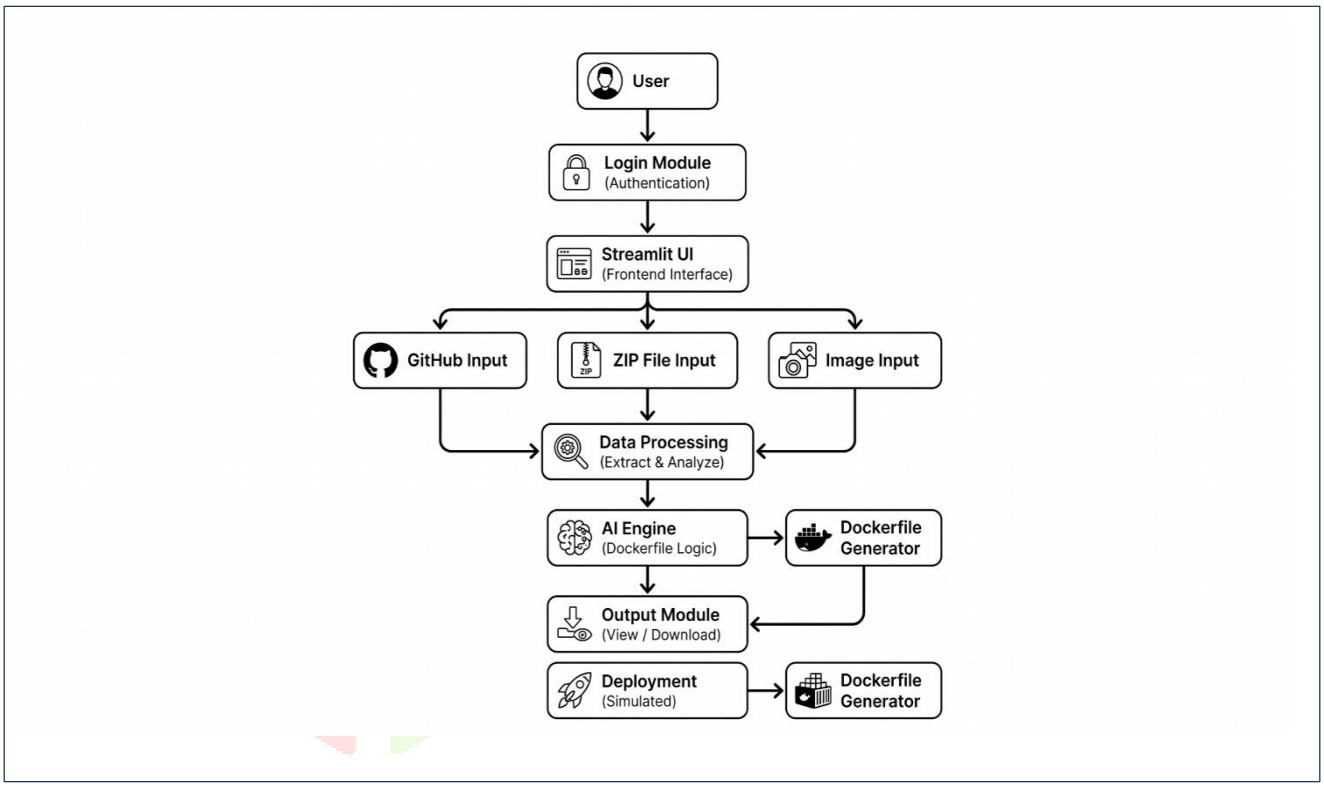


Fig 1: System Architecture of the Smart Deployment Assistant System

V. RESULT AND DISCUSSION

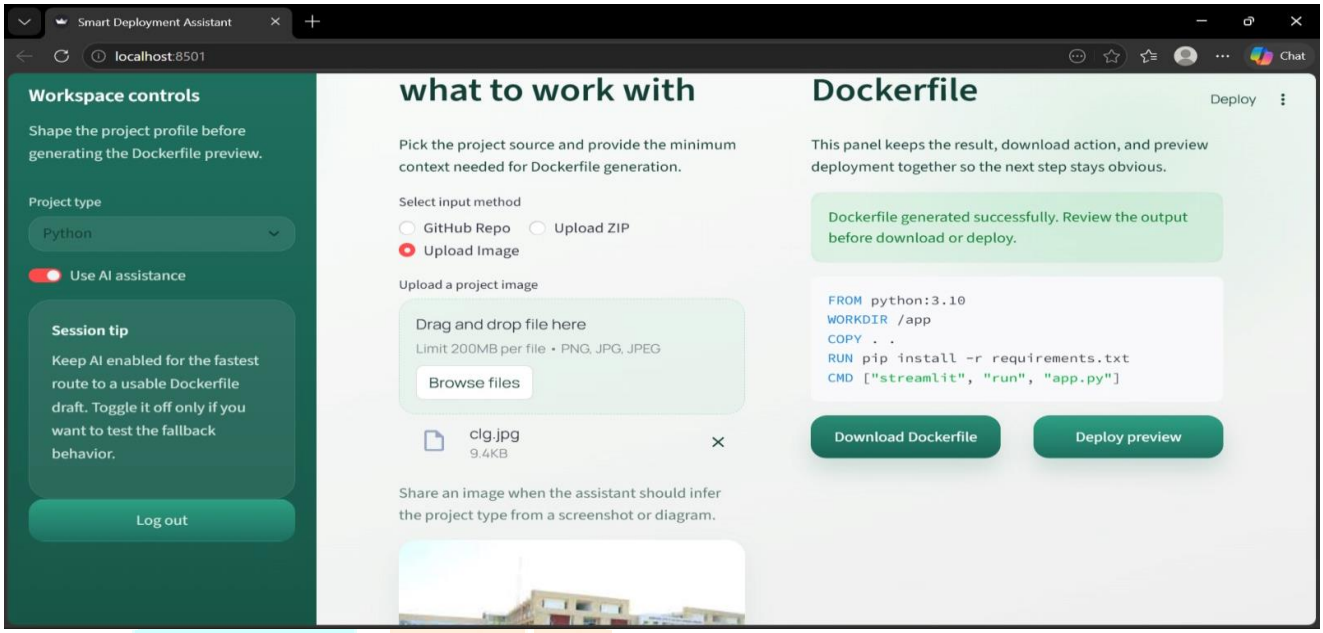
5.1 Result Analysis

The Smart Deployment Assistant was built and tested using multiple input formats, including GitHub repositories, ZIP files, and images. The system was evaluated based on its ability to generate valid Dockerfiles, reduce manual effort, and improve deployment efficiency. The system successfully generated Dockerfiles for different project types, including Python, Node.js, and Java. When valid input was provided, the AI module created Dockerfiles with essential instructions, including base image selection, dependency installation, and execution commands.

For GitHub repository inputs, the system accurately interpreted repository details and generated appropriate Dockerfiles. For ZIP file inputs, the system extracted project files and successfully identified the required dependencies. For image-based inputs, the system performed basic interpretation; however, the accuracy was comparatively lower due to limited contextual information.

The generated Dockerfiles were displayed to the user, who could download them for further use. Additionally, the deployment simulation feature demonstrated how the Dockerfile could be applied in a real-world environment.

5.2 USER INTERFACE



VI. FUTURE SCOPE

1. Advanced AI Integration

The current system uses rule-based logic. In the future, it can be upgraded using advanced Generative AI or machine learning models that can better understand complex project structures and generate more accurate Dockerfiles.

2. Support for Additional Technologies

The system can be further enhanced to include support for a wider range of programming languages and frameworks, such as:

- Django, Flask
- React, Angular
- .NET, PHP

This will make the tool more versatile and widely applicable.

3. Real-Time Cloud Deployment

The deployment module can be enhanced to perform actual deployment on cloud platforms like:

- AWS
- Microsoft Azure
- Google Cloud

This will enable users to deploy applications directly from the system.

VI. CONCLUSION

The Smart Deployment Assistant uses Generative AI to automate Dockerfile creation, making application deployment easier and faster, especially for beginners. By combining a simple interface with tools like Streamlit and Docker, the system reduces manual effort and errors while improving efficiency. Although it has some limitations, it provides a strong base for future enhancements and highlights the importance of AI-driven automation in modern DevOps workflows.

VII. REFERENCES

[1] D. Merkel, “Docker: Lightweight Linux Containers for Consistent Development and Deployment,” Linux Journal, no. 239, 2014.

[2] Docker Inc., Docker Documentation, Available: https://docs.docker.com/

[3] Streamlit Inc., Streamlit Documentation, Available: https://docs.streamlit.io/

[4] GitHub Inc., GitHub Documentation, Available: https://docs.github.com/ C. Pahl, “Containerization and the PaaS Cloud,” IEEE Cloud Computing, vol. 2, no. 3, pp. 24–31, 2015.