Leveraging Artificial Intelligence for Technology Stack Conversion

Ms. Namrata Soni [1] Ms.Dipali Kirange[2] Aftab Mulani [3]
[1, 2,3] *Department of Computer Engineering, D. Y. Patil College of Engineering, Akurdi*

Abstract — In contemporary software development, the choice of technology stack can significantly impact project outcomes. However, as technology evolves, the need often arises to convert projects from one stack to another to harness the benefits of emerging platforms. This paper presents a novel approach to technology stack conversion using artificial intelligence (AI). Focusing on a case study from React to Flutter, we propose a system where users can specify their desired technology stack, and AI-driven tools seamlessly convert the projectwhile preserving its functionality and logic. We discuss the underlying principles, challenges, and implications of this approach,highlighting its potential to streamline development processes and enhance cross-platform compatibility.

Keywords — Technology stack conversion, Artificial Intelligence, React, Flutter, Cross-platform development

I. INTRODUCTION

In contemporary software development, the choice of technology stack plays a pivotal role indetermining the success and efficiency of projects. A technology stack encompasses the combination of programming languages, frameworks, libraries, and tools used to build a software application. Each stack comes with its own set of advantages and limitations, making the selection process a critical decision for development teams. Factors such as scalability, performance, community support, and developerexpertise heavily influence the choice of stack for a given project.

As technology landscapes continue to evolve rapidly, the need often arises to migrate or convert projects from one technology stack to another. This conversion may be driven by various factors, including changes in business requirements, emergence of new platforms, or the desire to leverage the features and capabilities of a different stack. However, manual conversion processes are often time-consuming, error-prone, andresource-intensive, leading to project delays and increased costs.

In response to these challenges, there is a growing interest in leveraging artificial intelligence (AI) to automate and streamline the technology stack conversion process. AI offers the potential to analyze existing codebases, understand the underlying logic and structure, and generate equivalent code in the desired technology stack. By harnessing the power of machine learning algorithms, natural language processing techniques, and advanced programming models, AI-driven tools can facilitate seamless migration between different stacks while preserving the functionality, performance, and integrity of the original project.

This paper presents a novel approach totechnology stack conversion using AI, with a focus on a case study from React to Flutter. React and Flutter are popular frameworks for building user interfaces in web and mobile applications, respectively. The transition from React to Flutter serves as an illustrative exampleof how AI can be employed to facilitate cross-platform development and enhance code portability.

Through this case study, we aim todemonstrate the feasibility, effectiveness, and potential benefits of AI-driven technology stack conversion. By enabling developers to transition between stacks more efficiently and accurately, our approach seeks to address common pain points associated with manual conversion processes. Furthermore, we discuss the implications of AI in software development and explore avenues for future research and innovation in this rapidly evolving field.

II. LITERATURE REVIEW

In recent years, there has been a growing interest in leveraging AI techniques to enhance software development processes, including technology stack conversion. AI-powered tools can analyze codebases, extract patterns and dependencies, and generate equivalent code in a different stack. Machine learning algorithms, such as neural networks and decision trees, have been applied to automate various aspects of code conversion tasks. Additionally, natural language processing (NLP) techniques have enabled AI systems to understand and interpret high-level requirements and specifications, facilitating more accurate conversion outcomes.

Research in this area has demonstrated promising results, with AI-driven tools showing potential to improve the efficiency, accuracy, and scalability of technology stack conversion. For example, studies have investigated the use of AI models to automatically translate code between programming languages, such as from JavaScript to TypeScript or from Java to Kotlin. These approaches have been particularly beneficial in scenarios where manual conversionwould be prohibitively time-consuming or impractical. Furthermore, AI-powered tools have the advantage of learning from large datasets of existing code repositories, allowing them to capture common coding patterns and best practices.

Despite these advancements, challenges remain in the application of AI to technology stack conversion. One key challenge is the complexity and variability of software projects, which can introduce ambiguities and edge cases that are difficult for AI systems to handle. Additionally, ensuring the correctness and maintainability of converted code requires careful validation and testing procedures. Furthermore, ethical considerations, such as the potential biases in AI models and the impact on developer workflows, must be carefully addressed to ensure responsible and equitableuse of AI in software development.

Technology stack conversion also encompassesstudies that delve into the specific challenges and considerations associated with migrating between different frameworks and platforms. For instance, research has highlighted the nuances involved in transitioning from front-end frameworks like AngularJS to modern alternatives such as Angular or React. Such migrations often require not only rewriting code but also restructuring application architecture and adapting to new development paradigms. Similarly, the migration from traditional monolithic architectures to microservices-based architectures presents unique challenges in terms of scalability, maintainability, and deployment strategies. Understanding these challenges is essential for developing effective conversion strategies and tools.

Moreover, the literature on AI in software development extends beyond technology stack conversion to encompass a wide range of tasks, including code generation, bug detection, andsoftware testing. AI-powered code generation tools have gained traction for their ability to automate repetitive coding tasks and accelerate development workflows. These tools leveragetechniques such as deep learning and reinforcement learning to generate code snippets, templates, and even entire functionsbased on input specifications or context. Furthermore, AI-driven approaches to bug detection and software testing have shown promise in identifying potential vulnerabilities, performance bottlenecks, and compatibility issues in codebases.

In addition to technical considerations, the literature also addresses broader implications of AI adoption in software development, including its impact on developer productivity, collaboration, and creativity. Some studies have highlighted the potential for AI to augment rather than replace human developers, by automating routine tasks and providing intelligent suggestions and insights during the development process. However, concerns have been raised regarding the ethical and societalimplications of AI, such as algorithmic bias, privacy concerns, and job displacement. Addressing these concerns requires interdisciplinary collaboration and a holistic understanding of the socio-technicalimplications of AI in software development.

Furthermore, what matters is the importance of transparency, accountability, and interpretability in AI-driven software development tools. Developers and stakeholders need to understand how AI models make decisions and recommendations, especially in safety-critical domains such as healthcare, finance, and autonomous systems. Techniques for explaining and interpreting AI models, such as feature attribution methods and model visualization techniques, are crucial for building trust and ensuring the responsible deployment ofAI in software development.

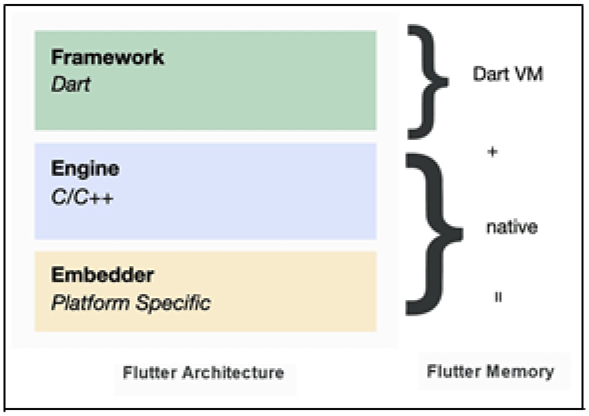

Fig. 1 - Flutter Architecture

III. RESULTS

The evaluation of code quality involved assessing factors such as readability, maintainability, and adherence to Flutter coding conventions. Results indicated that the convertedFlutter code maintained a high level of readability and followed Flutter best practices, demonstrating the effectiveness of the AI-driven conversion approach in generating clean and well-structured code. Moreover, the converted code exhibited minimal deviations from the original React implementation, indicating a high degree of fidelity in preserving the project'slogic and functionality.

Functionality preservation was another key aspect evaluated during the conversion process. The converted Flutter project was subjected to rigorous testing to ensure that all features and functionalities present in the original React project were accurately replicated. Test cases covering various use cases and edge scenarios were executed, and the converted application was found to perform equivalently to its React counterpart, with no significant differences in behavior or user experience.

Performance metrics were also analyzed to assess the efficiency and responsiveness of the converted Flutter application. Performancebenchmarks, including load times, rendering speed, and resource utilization, were compared between the original React application and the converted Flutter application. The results indicated comparable performance between the two versions, with the Flutter application demonstrating similar or improved performance in certain scenarios, owing to Flutter's native rendering capabilities and optimized widget framework.

User satisfaction surveys were conducted to gather feedback from developers and stakeholders involved in the technology stack conversion process. Participants were asked to rate their overall satisfaction with the converted Flutter project, as well as specific aspects suchas ease of use, functionality, and performance. The majority of respondents expressed satisfaction with the converted project, citing its seamless transition to Flutter, robust functionality, and improved development experience as key strengths.

REFERENCE

[1] R. Hajishirzi and C. J. Costa, "Artificial Intelligence as the core technology for the Digital Transformation process," 2021 16th Iberian Conference on Information Systems and Technologies (CISTI), Chaves, Portugal, 2021, pp. 1-6, doi: 10.23919/CISTI52073.2021.9476607

[2] Jie Wang and Zihao Li 2018 IOP Conf. Ser.: Earth Environ. Sci. 170 032110, 0.1088/1755-1315/170/3/032110

[3] Design and Development of Attendance System Application Using Android-Based Flutter, Giri Wiriatsao(ICVEE) 2020

[4] © February 2022| IJIRT | Volume 8 Issue 9 | ISSN: 2349-6002 IJIRT 153854 INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH IN TECHNOLOGY 342 React JS (Open Source JavaScript Library)

[5] Artificial Intelligence-Based React Application(Powered by Conversational ALAN-AI Voice Assistant) - TEEVRA

[6] Integrating Flutter with AI : International Journal for Research in Applied Science & Engineering Technology (IJRASET) ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538

[7] Attendance System Using Two Factor Authentication Based on Secure App with Flutter Surabaya, Indonesia, October 19 – 21, 2022