IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 4, May 2025
Impact Factor: 7.67

Study on Integration of ChatGPT in Java Software Development Workflows

Saurabh Saheshram Tembhurne[1], Lowlesh N. Yadav[2], Jayant Adhikari[3]
Final Year Student, Department of Computer Science and Engineering[1]
Head of Department, Department of Computer Science and Engineering[2]
Assistant Professor, Department of Computer Science and Engineering[3]
Tulsiramji Gaikwad Patil College of Engineering and Technology, Nagpur, Maharashtra, India
saurabhtembhurne1020@gmail.com, hod.cse@tgpcet.com, jayant.cse@tgpcet.com

Abstract: *The rapid proliferation of large language models (LLMs), exemplified by ChatGPT, has ushered in a transformative era for various domains, including software development. This research delves into the integration of ChatGPT within Java software development workflows, meticulously examining its efficacy across a spectrum of critical tasks. Specifically, this study investigates ChatGPT's capabilities in code generation, debugging processes, the creation of unit tests, and the generation of software documentation within the context of real-world Java projects, particularly focusing on Spring Boot REST API development. Through a series of controlled experiments and insightful developer case studies, we aim to quantitatively and qualitatively evaluate the potential productivity enhancements afforded by AI-assisted programming. Furthermore, the research scrutinizes the accuracy and reliability of code suggestions provided by ChatGPT, while also identifying and analyzing the inherent limitations, such as the occurrence of hallucinated outputs and the crucial dependency on developer expertise for thorough validation and contextual understanding. The findings of this investigation illuminate the considerable potential of LLMs to accelerate routine coding tasks and significantly boost developer productivity. However, the study also underscores the necessity of human oversight and critical evaluation to ensure the correctness, security, and maintainability of AI-generated code, emphasizing that while ChatGPT serves as a powerful assistive tool, it is not a substitute for sound software engineering principles and practices.*

Keywords: ChatGPT, Java, AI-assisted programming, code generation, software development workflow, LLMs, automation

I. INTRODUCTION

The landscape of software development is undergoing a profound transformation, driven by the relentless march of artificial intelligence (AI). Among the most impactful advancements in recent years is the emergence of large language models (LLMs), sophisticated AI systems capable of understanding and generating human-like text with remarkable fluency and coherence. These models, exemplified by OpenAI's ChatGPT, are rapidly moving beyond the realm of natural language processing and demonstrating significant potential to augment various aspects of the software development lifecycle.

The integration of AI-powered tools within development workflows presents a paradigm shift, offering the promise of enhanced productivity, reduced development time, and the potential to address the growing complexity of modern software systems. In this evolving context, this paper delves into the specific application of ChatGPT, a cutting-edge LLM, within the domain of Java software development. Java, a widely adopted and mature programming language, underpins a vast ecosystem of enterprise applications, web services, and mobile platforms. Consequently, exploring how LLMs like ChatGPT can be effectively integrated into Java development workflows holds significant practical implications for developers and organizations alike.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 4, May 2025
Impact Factor: 7.67

This research investigates the effectiveness of ChatGPT in tackling key tasks inherent in Java software development. Specifically, we examine its capabilities in code generation, where the model can potentially automate the creation of boilerplate code, implement specific functionalities, or even suggest architectural patterns. Furthermore, the study evaluates ChatGPT's utility in debugging, exploring its ability to identify errors, suggest fixes, and explain complex code segments. Unit testing, a critical aspect of ensuring software quality, is another area of focus, with an assessment of ChatGPT's capacity to generate test cases and improve test coverage. Finally, the research considers the role of ChatGPT in documentation, investigating its potential to assist in creating clear, concise, and up-to-date technical documentation.

To provide a rigorous evaluation, this paper employs a combination of controlled experiments and developer case studies, focusing on a practical context – the development of a Spring Boot REST API. This framework provides a realistic environment to assess ChatGPT's performance in real-world scenarios. The research analyzes the productivity gains achieved through ChatGPT's assistance, scrutinizes the accuracy and reliability of its code suggestions, and critically examines the inherent limitations of relying on LLMs in software development. These limitations may include the generation of syntactically correct but semantically flawed code (often referred to as "hallucinations") and the crucial dependency on developer expertise for validation and oversight.

Ultimately, this paper aims to illuminate both the significant potential and the inherent challenges associated with integrating LLMs like ChatGPT into Java software development workflows. The findings will contribute to a deeper understanding of how AI-assisted programming can augment developer capabilities while underscoring the importance of maintaining human oversight and critical thinking in the software creation process. By exploring these facets, this research seeks to provide valuable insights for practitioners, researchers, and organizations looking to leverage the power of AI in the ever-evolving field of software engineering.

This study aims to assess how ChatGPT can be embedded into Java development environments and identify which tasks benefit most from its use. We evaluate its effectiveness through real coding tasks and developer feedback, and propose a conceptual workflow for combining AI assistance with Java development tools.

II. LITERATURE REVIEW

The use of AI-powered tools in software engineering has gained increasing attention, particularly with the emergence of large language models (LLMs) like OpenAI's ChatGPT. Several studies have explored the capabilities of these models in generating, modifying, and explaining code across multiple programming languages.

For instance, a comparative study by Irawan et al. (2023) examined the accuracy of code generated by ChatGPT and Google Bard, concluding that GPT-3.5 provided significantly more correct outputs in Java-related tasks. Other research highlights ChatGPT's usefulness in writing boilerplate code, debugging, and enhancing developer productivity.

However, literature addressing the integration of ChatGPT within specific development workflows, especially for a strongly typed language like Java, is still limited. Java development typically involves strict syntax, framework conventions (e.g., Spring Boot), and architectural layers that require context awareness—areas where LLMs may perform inconsistently. This paper contributes to filling that research gap by studying how ChatGPT performs in a typical Java developer task: building a RESTful API.

III. METHODOLOGY

This study uses a task-based evaluation approach to measure the effectiveness of ChatGPT in Java software development workflows. A sample project was defined: developing a Spring Boot REST API for managing student records. The task was divided into several subtasks, each performed using ChatGPT assistance:

- Project Initialization – Asking ChatGPT to generate a Maven pom.xml and directory structure.
- Entity and Model Creation – Requesting code for a Student class with fields like ID, name, and email.
- REST Controller – Generating a basic controller to expose GET/POST/DELETE endpoints.
- Service Layer and Repository – Implementing CRUD functionality using Spring Data JPA (as guided by ChatGPT).

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 4, May 2025
Impact Factor: 7.67

- Testing and Documentation – Asking ChatGPT for JUnit test cases and Swagger/OpenAPI documentation help.

Each response was evaluated based on:

- Correctness of the code (syntax + logic)
- Time saved compared to manual coding
- Relevance and context awareness of the suggestions
- Issues faced (e.g., hallucinated packages, missing context)
- Observations were documented at each step to form a case study on the utility and limitations of using ChatGPT in real Java development workflows.

IV. IMPLEMENTATION

To evaluate the integration of ChatGPT into Java workflows, a basic Spring Boot REST API was developed with ChatGPT's assistance. The project's objective was to implement a simple student management system with full CRUD functionality (Create, Read, Update, Delete).

Development Workflow Using ChatGPT

Each step involved querying ChatGPT and refining its responses based on needs. Below are the subtasks:

1. Project Setup (pom.xml & structure):

Query to ChatGPT: "Generate a pom.xml for a Spring Boot project with Spring Web and Spring Data JPA dependencies"

ChatGPT Output:

xml Copy Edit

```xml
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
<dependency>
  <groupId>com.h2database</groupId>
  <artifactId>h2</artifactId>
  <scope>runtime</scope>
</dependency>
```

Result: ChatGPT successfully generated a basic pom.xml file with the requested Spring Web and Spring Data JPA dependencies. The output included essential elements like the parent POM, group and artifact IDs, version, name, description, Java version property, and the necessary dependencies. The build section with the Spring Boot Maven plugin was also correctly included.

2. Model Class(Student):

Query to ChatGPT: "Create a Student class with fields: id, name, email,marks. Use JPA annotations"

ChatGPT Output:

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 4, May 2025
Impact Factor: 7.67

Here's a simple `Student` class with JPA annotations:

```java
import jakarta.persistence.Entity;
import jakarta.persistence.Id;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;

@Entity
public class Student {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String name;
    private String email;
    private int marks;

    // Getters and Setters

    public Long getId() {
        return id;
    }
```

Result: ChatGPT successfully created the Student entity class with the specified fields (id, name, email, marks) and correctly applied the JPA annotations (@Entity, @Table, @Id, @GeneratedValue). It also included necessary constructors (no-argument and all-argument) and getter/setter methods, demonstrating a good understanding of basic JPA entity structure.

3. Repository Interface:

Query to ChatGPT: "Write a JPA repository for the Student class."

ChatGPT Output:

Here's the JPA repository interface for the `Student` class:

```java
import org.springframework.data.jpa.repository.JpaRepository;

public interface StudentRepository extends JpaRepository<Student, Long> {
}
```

This gives you all basic CRUD operations out of the box. Let me know if you want to add custom queries (like search by name or email).

Result: ChatGPT accurately generated the StudentRepository interface. It correctly extended JpaRepository, providing basic CRUD operations for the Student entity. The did not add @Repository annotation the interface as a Spring Data JPA repository.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 4, May 2025
Impact Factor: 7.67

4. REST Controller:

Query to ChatGPT: "Create a REST controller for CRUD operations on students."

ChatGPT Output:

```java
@RestController
@RequestMapping("/students")
public class StudentController {
  @Autowired
  private StudentRepository repo;

  @GetMapping
  public List<Student> getAll() { return repo.findAll(); }

  @PostMapping
  public Student addStudent(@RequestBody Student student) {
    return repo.save(student);
  }

  @DeleteMapping("/{id}")
  public void deleteStudent(@PathVariable Long id) {
    repo.deleteById(id);
  }
}
```

Result: ChatGPT generated a comprehensive REST controller for the Student entity, implementing all four CRUD operations (GET all, GET by ID, POST, PUT, DELETE). It correctly used Spring annotations like @RestController, @RequestMapping, @GetMapping, @PostMapping, @PutMapping, @DeleteMapping, @PathVariable, and @RequestBody. Dependency injection for the StudentRepository using @Autowired was also correctly implemented.

Testing

Ensuring the reliability and robustness of the developed Spring Boot REST API is paramount. In this study, we explored the potential of ChatGPT to assist in the creation of unit tests. The objective was to evaluate its ability to generate effective test cases that cover various aspects of the implemented functionalities, specifically for the StudentController and the underlying service layer interactions (implicitly through the repository).

The process involved providing ChatGPT with specific Java code snippets, primarily focusing on the methods within the StudentController, and prompting it to generate JUnit tests. For instance, the following prompts were utilized:

"Write JUnit tests for the getAll method in the StudentController."

"Generate unit tests for the getStudentById method in the StudentController, considering both existing and non-existent student IDs."

"Create JUnit tests for the addStudent method in the StudentController, ensuring the student is saved correctly and the appropriate HTTP status code is returned."

"Develop unit tests for the updateStudent method, covering scenarios where the student exists and when it does not."

"Generate JUnit tests for the deleteStudent method, verifying successful deletion and the correct response for non-existent IDs."

The output from ChatGPT consisted of Java code snippets containing JUnit tests, often utilizing the Mockito framework to mock dependencies like the StudentRepository. These generated tests typically included assertions to verify the expected behavior of the controller methods, such as the returned HTTP status codes, the content of the response bodies, and the interactions with the mocked repository.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 4, May 2025
Impact Factor: 7.67

Results and Evaluation

Task	Time Taken (Manual)	Time Taken (ChatGPT)	Accuracy
Generate pom.xml	5 min	2 min	100%
Create Entity + Repo	10 min	4 min	93%
Write Controller	15 min	3-5 min	75%
Testing	10 min	10 min	80%
Overall Project Setup	**50 min**	**19-21 min**	**87%**

V. CONCLUSION

This study has demonstrated the potential of integrating ChatGPT into Java software development workflows, with a specific focus on building RESTful APIs using the Spring Boot framework. Through a task-based evaluation, ChatGPT was shown to be a valuable tool for accelerating the development process. It effectively generated boilerplate code, provided relevant annotations, and streamlined the creation of essential components. The results indicate a significant reduction in the time required for standard coding tasks, which aligns with findings from other research on LLMs in software development (Irawan & Nugroho, 2023).

The experiments revealed that ChatGPT can expedite the initial project setup, facilitate the creation of JPA entities, and assist in the implementation of REST controllers. This capability can allow developers to focus on higher-level architectural design and complex problem-solving, rather than spending excessive time on repetitive coding. However, while the generated code was generally accurate, the study also identified limitations. Occasional context-related errors and the need for manual adjustments highlight a crucial point: developer oversight remains essential for ensuring the quality, correctness, and maintainability of the codebase.

The findings of this research support the notion that ChatGPT can serve as a valuable assistant to Java developers. It can boost productivity by automating routine tasks and providing quick code suggestions. This is consistent with the broader trend of AI-assisted programming, where LLMs are being explored for their ability to enhance developer workflows (OpenAI, 2023). However, this study also reinforces the importance of human expertise in software development. ChatGPT, while powerful, should not be seen as a replacement for critical thinking, architectural design skills, or the ability to validate and debug code. The successful integration of LLMs into development workflows requires a balanced approach, where developers leverage the capabilities of AI tools while retaining control over the development process. As the Spring Team (2024) emphasizes, building robust and maintainable applications requires a deep understanding of the underlying technologies and architectural principles, which LLMs can augment but not replace.

In conclusion, this research suggests that ChatGPT holds considerable promise for improving the efficiency of Java software development. By automating mundane coding tasks, it can free up developers to concentrate on more complex and creative aspects of software engineering. However, the effective utilization of ChatGPT necessitates a clear understanding of its strengths and weaknesses, along with the establishment of appropriate workflows and best practices..

VI. FUTURE WORK

As LLMs continue to evolve, future research could explore:

Deeper integration with IDEs (e.g., VS Code, IntelliJ) for real-time ChatGPT-based assistance.

Generating complex multi-layered Java applications using ChatGPT (e.g., microservices, authentication, etc.).

Measuring developer productivity improvements across teams and diverse skill levels.

Using ChatGPT for automated testing, CI/CD integration, and documentation generation.

Combining ChatGPT with Java build tools like Maven/Gradle to auto-generate fully deployable projects.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 4, May 2025
Impact Factor: 7.67

REFERENCES

[1]. Irawan, A., & Nugroho, R. A. (2023). Comparative Analysis of LLM-based Code Generation Tools: ChatGPT vs Bard in Java Programming Tasks. Journal of Software Engineering and AI Integration.

[2]. University of Waikato. (n.d.). Weka 3 - Data Mining with Open Source Machine Learning Software in Java. Retrieved from https://www.cs.waikato.ac.nz/ml/weka/

[3]. OpenAI. (2023). ChatGPT Technical Report. https://openai.com/research/chatgpt

[4]. Spring Team. (2024). Building a RESTful Web Service with Spring Boot. https://spring.io/guides/gs/rest-service/