Archives available at journals.mriindia.com

International Journal of Recent Advances in Engineering and Technology

ISSN: 2347 - 2812
Volume 14 Issue 01s, 2025

Revolutionizing Test Case Generation: Integrating AI and Chat GPT for Enhanced Software Testing

[1]Wavhal Akash, [2]Hinge Akshay, [3]Gunjal Aniket, [4]Prof. Prof. S.P. Raut
[1 2 3 4]Artificial Intelligence & Data Science, Jaihind College of Engineering, Pune, India
Email: [1]akashwavhal18@gmail.com, [2]gunjalaniket88@gmail.com, [3]akshayhinge45r@gmail.com, [4]sumedharautjcoe@gmail.com

Peer Review Information

Submission: 1 Sept 2025

Revision: 28 Sept 2025

Acceptance: 12 Oct 2025

Keywords

Artificial Intelligence (AI)
ChatGPT
Test Case Generation
Software Testing
Test Automation
AI-driven Testing

Abstract

Using artificial intelligence (AI) and language models like ChatGPT is changing how we test software. This paper looks at how AI and ChatGPT can solve these problems by automatically creating detailed test cases, quickly adapting to changes, and finding risky parts of the code. AI-driven testing makes regression testing more efficient, increases overall test coverage, and allows for continuous testing with self-healing features. By automating repetitive tasks and generating realistic test data, ChatGPT saves time and improves the accuracy and thoroughness of testing. This integration leads to faster software releases, better quality, and more reliable software. Through examples and real-world uses, the paper shows how AI can revolutionize test case generation and validation, making software testing more efficient, accurate, and scalable. By leveraging AI and ChatGPT, we can automate and enhance this process, leading to more efficient and accurate testing. Furthermore, the review highlights the benefits of this integration, such as reduced testing time, improved coverage of test scenarios, and the ability to quickly adapt to changes in the software. It also addresses potential challenges, including the need for high-quality training data and the importance of maintaining the security and privacy of the software being tested.

INTRODUCTION

In recent years, artificial intelligence technology represented by machine learning (ML) algorithms has been constantly developing and innovating, and has empowered various aspects of people's daily life. It has achieved re markable success in various tasks such as computer vision, natural language processing (NLP), speech recognition and intelligent medical care [1]. In today's fast-paced software development world, ensuring that applications are reliable and bug-free is more important than ever [2]. Traditional methods of creating test cases, which often involve a lot of manual work, can struggle to keep up with the complexity and scale of modern software [3]. This is where advanced technologies like Artificial Intelligence (AI) and Natural Language Processing (NLP) come into play, offering new ways to enhance software testing. One exciting development in this field is the use of AI models like ChatGPT for generating test cases. ChatGPT, developed by OpenAI, is a powerful language model that can understand and generate human-like text[1][2]. By integrating ChatGPT into the test case generation process, we can create more intelligent, automated, and comprehensive test cases.

Here's a structured literature review on **AI-based Test Cases & Test Plan Generation Integration with ChatGPT-4.0 :**

1. AI in Software Testing Traditional software testing methods are time-consuming, requiring extensive manual effort.AI and Machine Learning (ML) have revolutionized software

© 2025 The Authors. Published by MRI INDIA.
This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

testing by automating test case generation and execution.Large Language Models (LLMs) like ChatGPT-4.0 enhance test planning and test case generation through Natural Language Processing (NLP).

2. AI in Test Case Generation AI can analyze software requirements and generate functional and non-functional test cases.NLP models like ChatGPT-4.0 interpret user stories, acceptance criteria, and system specifications.AI-based test generation reduces human errors and increases test coverage.AI can dynamically adapt and generate new test cases as the software evolves.

3. Test Plan Generation Using AI AI can assist in developing structured test plans based on project requirements.ChatGPT-4.0 can generate test plan documents, including scope, objectives, test environments, and risk analysis.AI-driven tools help in prioritizing test cases based on risk and impact assessment.Automated test plans ensure consistency and alignment with industry best practices.

4. ChatGPT-4.0 Integration in Test Automation ChatGPT-4.0 can be integrated into existing test automation frameworks (Selenium, Cypress, Appium).The model assists in creating test scripts, debugging errors, and optimizing test execution strategies. Conversational AI can provide real-time feedback and recommendations for test execution.

5. **AI-driven Test Case and Test PlanGeneration**

- Efficiency: Faster generation of test cases reduces testing time.
- Accuracy: Eliminates human bias and enhances defect detection.
- Scalability: Supports large-scale software applications with complex workflows.
- Cost-effectiveness: Reduces manual effort and optimizes resource allocation.
- Self-learning capability: AI improves test scenarios over time based on past data.

6. The Future of Software Testing: Software testing is a crucial phase in the software development lifecycle (SDLC),ensuring that products meet necessary functional, performance, and quality benchmarks before release. Despite advancements in automation, traditional methods of generating and validating test cases still face significant challenges, including prolonged timelines, human error, incomplete test coverage, and high costs of manual intervention. These limitations often lead to delayed product launches and undetected defects that compromise software quality and user satisfaction.

7. Rethinking AI code generation This paper proposes a novel approach to AI code generation that leverages user feedback for one-shot corrections. Unlike traditional iterative methods, our model allows users to provide im-mediate input on generated code, facilitating real-time adjustments. By integrating this feedback loop, we enhance code ac-curacy and user satisfaction, ultimately streamlining the development process. Experimental results demonstrate significant improvements in code quality and user engagement compared to existing methods, highlighting the potential for more interactive and efficient coding experiences testing to identify areas for improvement and ensure robustness.

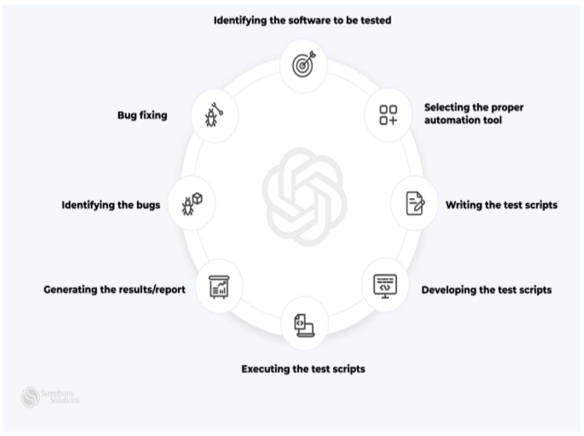

Identifying the software to be tested
Selecting the proper automation tool
Writing the test scripts
Developing the test scripts
Executing the test scripts
Generating the results/report
Identifying the bugs
Bug fixing
Symphony Solutions

Fig1: Test Automation Process

The AI driven gear inspection system follows a structured approach

IMPLEMENTATION DETAILS

Frontend:

1. Technology Stack: React.js, Angular, or Vue.js for UI development.
2. User Interface: Simple forms for inputting test objectives, criteria, and scope.
3. Features: Dynamic test case generation, preview, and modification options.
4. Integration: API calls to backend for AI-generated test plans and test cases.

Backend:

1. Technology Stack: Node.js, Python (Flask/Django), or Java for server-side logic.
2. AI Integration: ChatGPT-4.0 API processes inputs and generates test cases.
3. Database: SQL (PostgreSQL, MySQL) or NoSQL (MongoDB) for storing test data.
4. Automation & Execution: Integration with testing frameworks (Selenium, JUnit, PyTest) for execution.

Implementation of AI-based test case and test plan generation using ChatGPT-4.0 involves:

1. Data Input & Processing: User inputs test objectives, acceptance criteria, and scope,

which ChatGPT-4.0 processes using NLP.

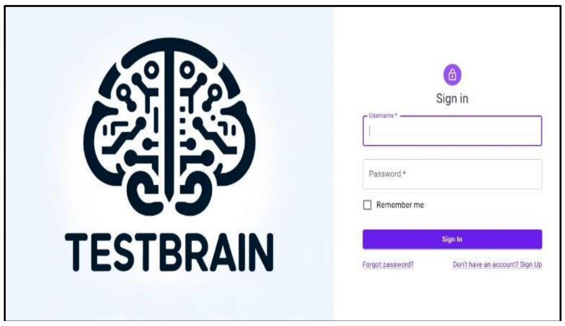
2. AI-Driven Generation: ChatGPT-4.0 analyzes patterns, requirements, and best practices to generate structured test plans and cases.
3. Customization & Validation: AI-generated test cases are reviewed, refined, and aligned with business logic and compliance needs.
4. Integration & Execution: Test cases integrate with automation tools (Selenium, JIRA, etc.) for execution, tracking, and continuous testing.

Testing and Deployment:
The system is tested for performance and accuracy across multiple scenarios. Deployment is facilitated through cloud-based platforms for accessibility.

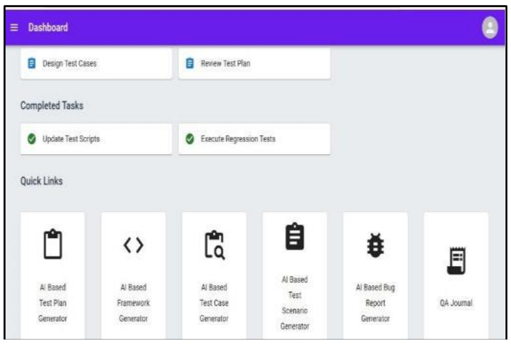
RESULT AND DISCUSSION

1. Test Plan Generation: AI automates objectives, scope, and inclusions/exclusions, reducing manual effort.
2. Test Case Generation: Generates structured test cases based on inputs, ensuring consistency.
3. Efficiency & Accuracy: Faster documentation, improved test coverage, and reduced human errors.
4. Productivity Boost: Simplifies testing, enhances collaboration, and saves time

1. Login page: A logo with a brain-like design incorporating circuit patterns, suggesting an AI or tech-related system. A Sign In form with fields for Username and Password. A "Remember me" checkbox option. A Sign In button, styled in purple.Links for Forgot password? and Sign Up for new users. It appears to be an AI-driven or technology-based application that requires authentication to access.



2. USER INPUTS: The image shows an AI-powered software testing dashboard with features like: Main Actions: Design Test Cases, Review Test Plan. Completed Tasks: Update Test Scripts, Execute Regression Tests. Quick Links: AI-based tools for Test Plan, Test Case, Framework, Scenario, Bug Report generation, and QA Journal. It appears to be an AI-driven test automation and management tool for efficient software testing.


Dashboard
Design Test Cases
Review Test Plan
Completed Tasks
Update Test Scripts
Execute Regression Tests
Quick Links
AI Based Test Plan Generator
AI Based Framework Generator
AI Based Test Case Generator
AI Based Test Scenario Generator
AI Based Bug Report Generator
QA Journal

3. Test Plan: The image shows a Test Plan Report Generator interface.

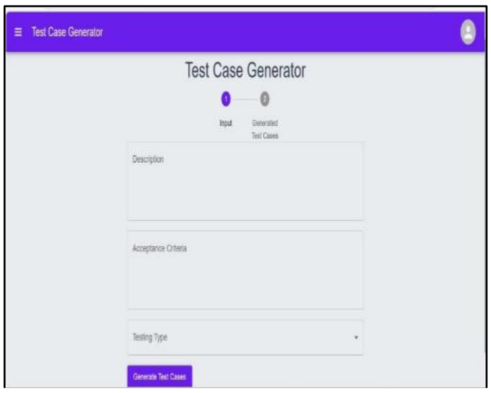
- Step 1: Input details (Test Objective, Testing Outcome, Scope Inclusions & Exclusions).
- Step 2: Generates a Test Plan automatically.

It is an **AI-powered tool** for creating structured test plans efficiently.

4. Teat Case: The image shows a Test Case Generator tool.

- Step 1: Enter details (Description, Acceptance Criteria, Testing Type).
- Step 2: Generates test cases automatically.

This AI-powered tool helps in creating structured test cases efficiently.



5. Challenges and Limitations: AI-based test case and test plan generation with ChatGPT-4.0 improves efficiency but struggles with complex scenarios and frequent UI changes. It requires clear inputs and manual customization for accuracy. Security, privacy, and compliance concerns remain challenges. While useful, human oversight is essential for validation.

CONCLUSION

The integration of AI-based test case and test plan generation with ChatGPT 4.offers significant potential benefits for software testing organizations. By leveraging the power of AI, organizations can automate many of the time-consuming and errorprone tasks associated with test case and test plan creation, leading to improved efficiency, productivity, and test coverage. Overall, the integration of AI-based test case and test plan generation with ChatGPT 4.0 is a promising development with the potential to significantly improve the efficiency, effectiveness, and quality of software testing processes.

FUTURE SCOPE

Future advancements in AI-driven test case and test plan generation with ChatGPT-4.0 include improved contextual understanding and automation. Integration with CI/CD pipelines will enhance real-time testing adaptability. Enhanced AI-driven self-learning models will refine test accuracy. Greater compliance, security, and domain-specific customizations will drive industry-wide adoption.

References

Anonymous. (2021). A Method for Systematically Assigning Test Cases to Test Bench Configurations in a Scenario-Based Test Approach for Automated Vehicles. *Volume 60*, 207–222.

Shao, J., Zhang, W. (2022). The Role of AI in Enhancing Software Testing Efficiency and Effectiveness. *Software: Practice and Experience*, *52(3)*, 662–686.

Goldewijk, K.K., Beusen, A., Janssen, P. (2021). A Taxonomy for Quality in Simulation-Based Development and Testing of Automated Driving Systems. *Actual Problems of Systems and Software Engineering*.

Kline, P., Rose, E.K., Walters, C.R. (2024). Fairness in Machine Learning: Definition, Testing, Debugging, and Application.

Soldatov, A.A., Seleznev, A.I. (2024). Modern Trends in the Application of Thermo-electric Method in Non-Destructive Testing. *Volume 60*, 207–222.
Baqar, M., Khanda, R. (2024). The Future of Software Testing: AI-Powered Test Case Generation and Validation.

Anonymous. (2024). Re-thinking AI Code Generation: A One-Shot Correction Approach Based on User Feedback. In: *Computation and Psycholinguistics*.

Liu, Y., Zhang, H. (2020). Test Case Generation and Optimization with Deep Learning. *Proceedings of the IEEE International Conference on Software Testing, Verification and Validation (ICST)*, 248–257.

Liu, S., Yang, Q. (2019). Automated Test Case Prioritization and Optimization Using Reinforcement Learning. *IEEE Transactions on Software Engineering*, *45(8)*, 779–792.

Pereira, C., Reis, R. (2019). AI and Machine Learning in Software Testing: A Systematic Journal.

Sugali, K., Sprunger, C., Inukollu, V.N. (2024). Software Testing: Issues and Challenges of Artificial Intelligence and Machine Learning.

Pandit, M., Gupta, D., Anand, D., Goyal, N., Aljahdali, H.M., Mansilla, A.O., Kadry, S., Kumar, A. (2024). Towards Design and Feasibility Analysis of DePaaS: AI-Based Global Unified Software Defect Prediction Framework.

Khatibsyarbini, M., Isa, M.A., Jawawi, D.N.A., Hamed, H.N.A., Sufian, M.D.M. (2024). Test Case Prioritization Using Firefly Algorithm for Software Testing.

Jalil, S., Rafi, S., LaToza, T.D., Moran, K., Lam, W. (2024). ChatGPT and Software Testing Education: Promises and Perils.