International Journal of Research and Applied Innovations (IJRAI)

| ISSN: 2455-1864 | www.ijrai.org | editor@ijrai.org | A Bimonthly, Scholarly and Peer-Reviewed Journal |

|| Volume 9, Issue 3, May-June 2026||

DOI:10.15662/IJRAI.2026.0903008

Automated Bug Detection and Auto Fix Generation by using Ml Model

Mrs. D Sangeetha[1], R Swathi[2], S Pavithra Sree[3], K Sinduja[4]

Assistant Professor, Department of Information Technology, Jaya Engineering College, Anna University, Chennai, Tamil Nadu, India[1]

UG Student, Department of Information Technology, Jaya Engineering College, Anna University, Chennai, Tamil Nadu, India[2,3,4]

Publication History: Received: 25.04.2026; Revised: 01.05.2026; Accepted: 03.05. 2026; Published: 09.05.2026.

ABSTRACT: This project is a developing an Intelligent Machine Learning based system for automated bug detection and auto fix generation in software applications. Debugging is one of the most time-consuming and challenging tasks in software development, especially for beginners. In this system, the user provides source code or error messages as input. The system analyzes the code using Machine Leaming and Natural Language Processing (NLP) techniques trained on large datasets of buggy and corrected code. Based on learned patterns, the model identifies the type of bug and generates appropriate fix suggestions. The system works similar to a chatbot, providing interactive and user-friendly debugging assistance. This intelligent approach reduces human effort. improves coding efficiency, and helps developers learn better programming practices. The proposed system is cost-effective, scalable, and suitable for real-time software development environments. It provides real-time feedback, enabling immediate correction of mistakes during development. The machine learning model continuously updates its knowledge base as new types of bugs are encountered.

The proposed system accepts source code or error messages as input from the user. It processes this input using advanced ML models trained on large datasets containing buggy code samples and their corresponding corrected versions. By learning patterns, structures, and common error types from historical data, the model is capable of identifying syntax errors, logical bugs, runtime exceptions, and semantic inconsistencies in the code. The integration of NLP techniques enables the system to interpret error messages and understand programming context, making it more adaptable across different programming languages and development environments.

KEYWORDS: Machine Learning. Automated Bug Detection, Auto Fix Generation Software Debugging, Large Language Model (LLM), Intelligent System, Code Analysis, Error Detection, Software Development, Pattern Recognition, Real-time Feedback, Code Correction Developer Productivity, Scalable System, Al-based Debugging. Bug Classification, Self-learning System

I. INTRODUCTION

Machine Learning is an advanced and influential domain of Artificial Intelligence that enables computer systems to learn from data and improve their performance without being explicitly programmed for every task. It focuses on the development of algorithms and models that can automatically identify patterns, analyze relationships, and make intelligent decisions based on past experiences and large volumes of data. Traditional software systems follow fixed rules defined by programmers, whereas machine learning systems adapt and evolve by learning from data inputs. This capability allows machine learning to handle complex and dynamic problems that are difficult to solve using conventional programming methods. Machine learning techniques are commonly classified into supervised learning, unsupervised learning, and reinforcement learning, each designed to address specific types of problems such as prediction, classification, clustering, and optimization. The increasing availability of big data, cloud computing, and high-performance hardware has significantly accelerated the growth and effectiveness of machine learning systems across various industries.

The domain of machine learning has become a key driver of innovation and automation in modern technology-driven environments. It is widely applied in real-world scenarios such as medical diagnosis, financial forecasting, fraud

| ISSN: 2455-1864 | www.ijrai.org | editor@ijrai.org | A Bimonthly, Scholarly and Peer-Reviewed Journal |

|| Volume 9, Issue 3, May-June 2026||

DOI:10.15662/IJRAI.2026.0903008

detection, recommendation systems, image and speech recognition, customer behavior analysis, and autonomous systems. By learning from historical data and continuously updating its models, machine learning enhances accuracy, efficiency, and decision making capabilities over time. Organizations leverage machine learning to gain valuable insights, reduce human effort, and improve operational performance. As data continues to grow exponentially, the importance of machine learning in extracting meaningful knowledge and supporting intelligent systems becomes increasingly significant. Consequently, machine learning stands as a foundational domain that supports digital transformation and plays a crucial role in shaping the future of intelligent technologies. Even small mistakes in code can lead to system failures, security vulnerabilities, and performance issues. Hence, there is a strong need for intelligent systems that can automatically analyze source code, identify potential bugs, and suggest effective fixes. Detecting bugs and fixing errors manually is a time-consuming process that requires significant developer effort and expertise.

II. LITERATURE SURVEY

Chintha Durga Ram Charan Tej proposed an automated software bug detection system using Machine Learning techniques such as Random Forest, Support Vector Machine (SVM), and Deep Neural Networks to improve software quality assurance. The study utilized Natural Language Processing (NLP) to analyze source code patterns and achieved improved precision, recall, and F1-score, making it effective for automated debugging.[1]

Bahtiar Imran et al. developed a machine learning model for automatic bug detection in Python code using syntactic analysis. Their approach used Abstract Syntax Tree (AST)-based feature extraction combined with a Random Forest classifier, achieving an accuracy of 86.67%. The model effectively analyzed structural code features such as functions, variables, and control statements, though it was limited to syntactic errors.[2]

Jiajun Sun and Fengjie Li conducted an empirical evaluation of Large Language Models (LLMs) in automated program repair (APR). Their study compared LLM-based approaches with traditional repair techniques and found that LLMs significantly improved repair success rates for syntax and semantic errors, while facing challenges in handling complex logical bugs and generalization.[3]

Islem Bouzenia and Premkumar Devanbu introduced an autonomous LLM-based agent called Repair Agent for program repair. This system used a finite-state machine and tool-based interaction to autonomously detect, analyze, and fix bugs. Experimental results showed that the agent successfully fixed a large number of real-world bugs, demonstrating the effectiveness of autonomous AI-driven software maintenance.[4]

Alike Saeed discussed the role of machine learning in automated bug detection within software quality assurance (QA). The study highlighted how ML models analyze source code, test data, and historical bug patterns to detect defects early, reduce manual effort, and improve software reliability and development efficiency.[5]

III. RESEARCH METHODOLOGY

3.1 PROBLEM IDENTIFICATION

Manual debugging in software development is a time-consuming and complex process. Developers often spend a significant amount of time identifying and fixing bugs, which reduces overall productivity. Traditional debugging tools are limited in their ability to understand the context of the code and often fail to detect logical and semantic errors. Additionally, these tools do not provide automatic fix suggestions, increasing the manual effort required. As software systems become more complex, the number of bugs also increases. This leads to higher development time, cost, and reduced code quality. Therefore, there is a need for an intelligent and automated bug detection system.

3.2 PPROBLEM SOLVING

To overcome these challenges, the proposed system uses machine learning techniques to automatically detect bugs in source code. The system analyzes code using advanced algorithms and identifies various types of errors such as syntax, logical, and runtime issues. It also generates context-aware fix suggestions, reducing the need for manual debugging. By learning from historical bug data, the model improves its accuracy over time. The system provides a user-friendly interface for easy interaction. This approach significantly reduces development time and effort. Overall, it enhances code quality, reliability, and developer productivity.

3.3 SYSTEM ARCHITECTURE

International Journal of Research and Applied Innovations (IJRAI)

| ISSN: 2455-1864 | www.ijrai.org | editor@ijrai.org | A Bimonthly, Scholarly and Peer-Reviewed Journal |

|| Volume 9, Issue 3, May-June 2026||

DOI:10.15662/IJRAI.2026.0903008

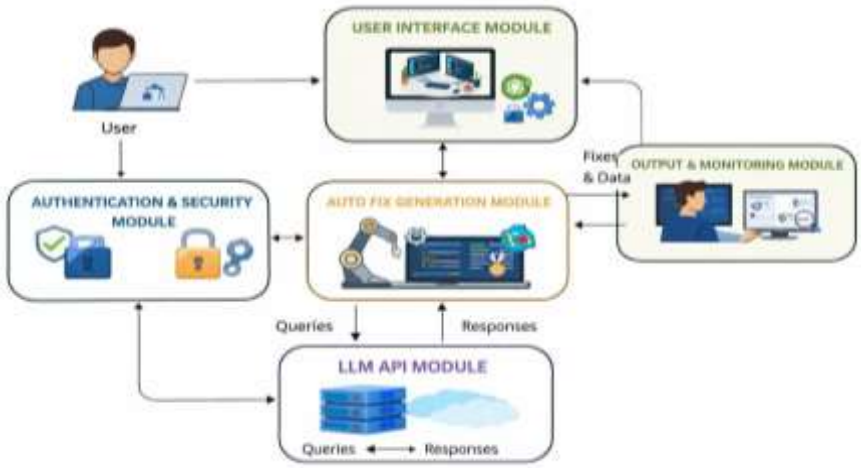

Fig.1 System Architecture for Error Detection and Auto fix generation

The proposed system architecture for automated bug detection and fixing integrates several modules to ensure efficient and secure operation. Initially, the user interacts with the system through the User Interface Module, where code or queries are submitted. The Authentication and Security Module verifies user credentials and ensures data protection. Once authenticated, the request is processed by the Auto Fix Generation Module, which analyzes the code, detects bugs, and generates fixes using machine learning and LLM-based techniques. This module communicates with the LLM API Module to send queries and receive intelligent responses for accurate bug identification and correction. The results are then forwarded to the Output and Monitoring Module, which evaluates, tracks, and displays the fixes along with performance insights. Finally, the processed output is returned to the user through the interface, completing the workflow and improving overall software quality with minimal manual effort. This Modules shows,

1.User Interface Module
2.Authentication &Security Module
3. Auto Fix Generation Module
4.LLM API Integration Module

User Interface Module
The User Interface (UI) Module acts as the primary point of interaction between the user and the system. It provides a secure login mechanism, allowing users to access the dashboard and utilize system features efficiently. The interface is designed to be intuitive and user-friendly, ensuring ease of use for both beginners and experienced developers.The module enables users to input source code through multiple options such as live code editor, copy-paste functionality, or file upload. It also allows users to select the desired programming language and initiate real-time manual analysis. Once the code is submitted, the UI displays detected bugs categorized by severity levels (critical, high, medium) along with detailed explanations and suggested fixes. Additionally, the module provides visual feedback such as alerts, notifications, and processing status to keep the user informed throughout the analysis process. It also supports features like applying fixes directly, clearing input, and viewing past analysis history. Overall, the UI module ensures smooth interaction, efficient workflow, and an enhanced user experience.

Authentication & Security Module
The Authentication and Security Module ensures that only authorized users can access the system. It verifies user identity through a secure login mechanism using username and password credentials. To enhance security, sensitive data such as passwords are protected using hashing and encryption techniques, preventing unauthorized access or data breaches.The module also manages user sessions securely using tokens, ensuring safe communication between the client and server. It implements HTTPS protocols to protect data transmission from external threats. Additionally, this module prevents unauthorized access, detects suspicious activities, and maintains overall system integrity, making the application reliable and secure for real-world usage. It protects sensitive data using hashing and encryption techniques, and manages sessions securely with tokens. The module also uses HTTPS protocols to ensure safe data transmission

International Journal of Research and Applied Innovations (IJRAI)

| ISSN: 2455-1864 | www.ijrai.org | editor@ijrai.org | A Bimonthly, Scholarly and Peer-Reviewed Journal |

|| Volume 9, Issue 3, May-June 2026||

DOI:10.15662/IJRAI.2026.0903008

and prevent unauthorized access. Overall, it maintains system integrity and provides a secure and reliable environment for users.

Auto Fix Generation Module

The Auto Fix Generation Module is responsible for automatically generating solutions for the detected bugs. After analyzing the uploaded code, the system identifies different types of errors such as syntax, logical, and runtime issues using machine learning techniques. Based on these detections, the module generates appropriate and context-aware fix suggestions.

This module leverages advanced models, including Large Language Models (LLMs), to improve the accuracy and relevance of the fixes. It ensures that the suggested corrections follow best coding practices and enhance code quality. The module reduces manual debugging effort by providing instant and reliable fixes, allowing developers to quickly resolve issues and improve productivity. It then generates context-aware fix suggestions with the help of advanced models like Large Language Models (LLMs), ensuring accuracy and adherence to coding best practices. This module minimizes manual debugging effort by providing quick and reliable fixes, ultimately improving code quality and developer productivity.

LLM API Integration Module

The LLM API Integration Module acts as a bridge between the system and external Large Language Model services. It sends the user's source code as a request to the API and receives detailed responses, including bug analysis and fix suggestions.

The module processes the received responses and forwards them to the appropriate system components, such as the auto-fix and output modules. It sends the user's source code to the API and receives detailed responses including error analysis and fix suggestions. The module then processes these responses and forwards them to relevant components like the auto-fix and output modules. By ensuring smooth communication and efficient data flow, it improves the overall intelligence, accuracy, and performance of the system.

VI. RESULTS

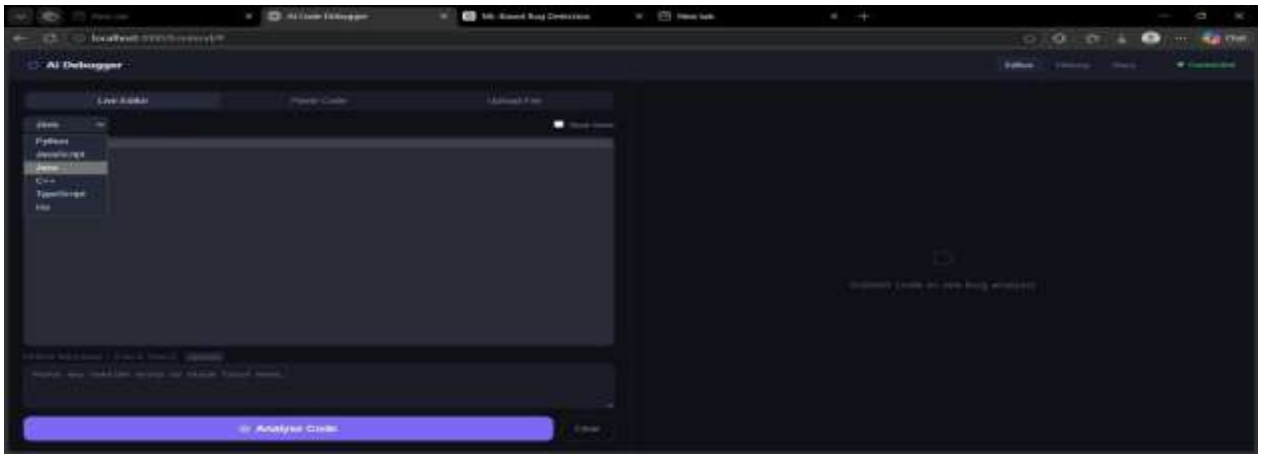

Fig.2 Home Page

This Fig.2 shows the user interface allows selecting a programming language and entering or uploading code for analysis. Initially, no output is shown until the user submits code for bug detection

International Journal of Research and Applied Innovations (IJRAI)

| ISSN: 2455-1864 | www.ijrai.org | editor@ijrai.org | A Bimonthly, Scholarly and Peer-Reviewed Journal |

|| Volume 9, Issue 3, May-June 2026||

DOI:10.15662/IJRAI.2026.0903008

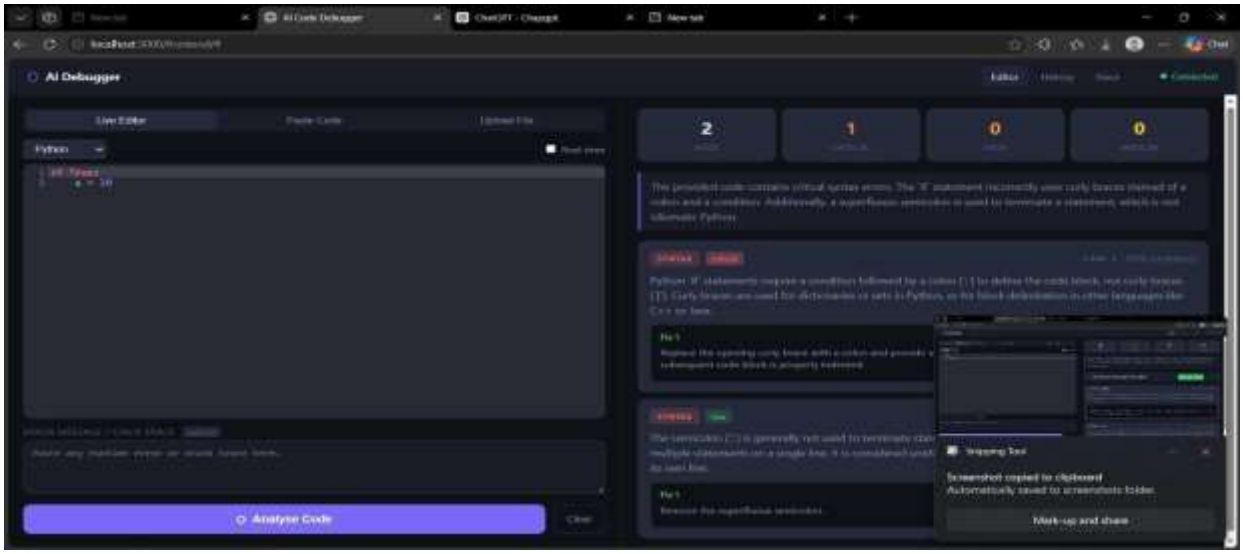

Fig.3 Live Editor Page

This Fig.3 shows the system analyzes Python code and detects syntax errors, showing bug count and severity levels. It also provides clear explanations and fix suggestions with high confidence.

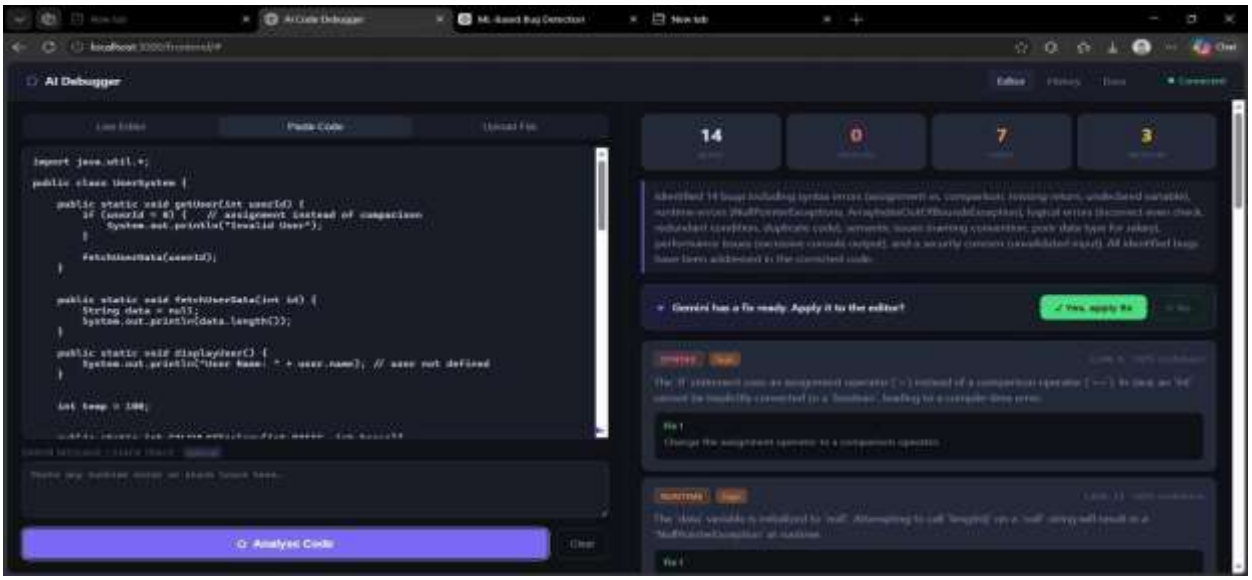

Fig.4 Paste Code Page

This Fig.4 shows the system identifies multiple bugs in Java code, including logical, runtime, and syntax issues.It categorizes errors by severity and allows users to apply automatic fixes directly.

International Journal of Research and Applied Innovations (IJRAI)

| ISSN: 2455-1864 | www.ijrai.org | editor@ijrai.org | A Bimonthly, Scholarly and Peer-Reviewed Journal |

|| Volume 9, Issue 3, May-June 2026||

DOI:10.15662/IJRAI.2026.0903008

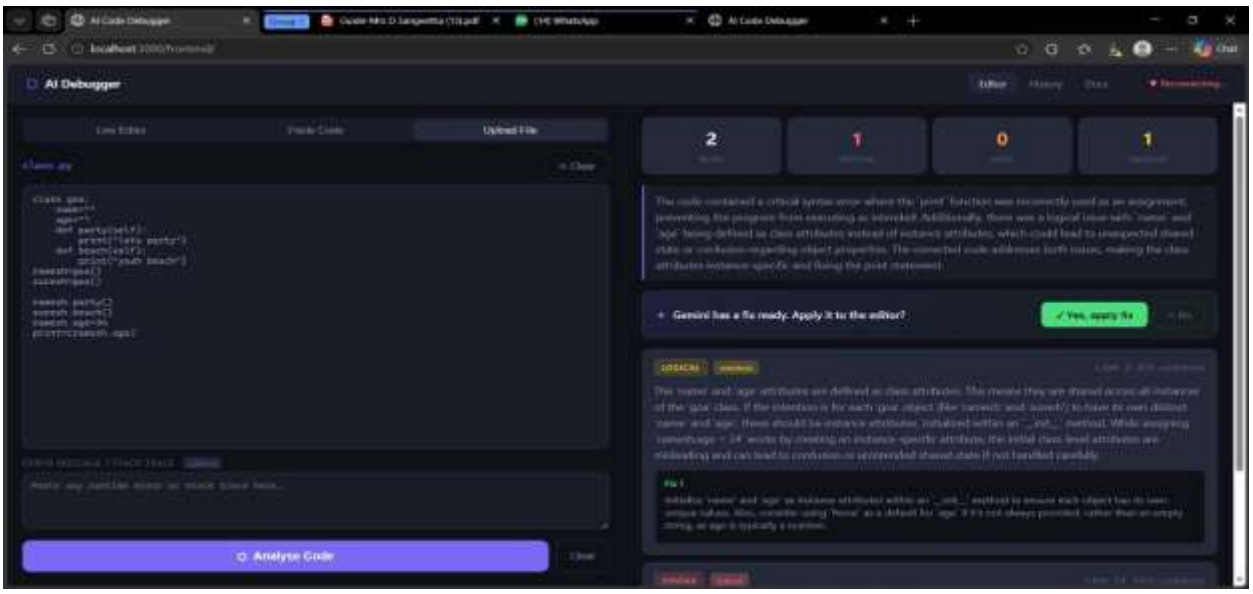

Fig.5 Upload File page

This Fig.5 shows the system generates intelligent fix suggestions for detected bugs and explains the issues.Users can review and apply fixes instantly, improving code correctness and quality.

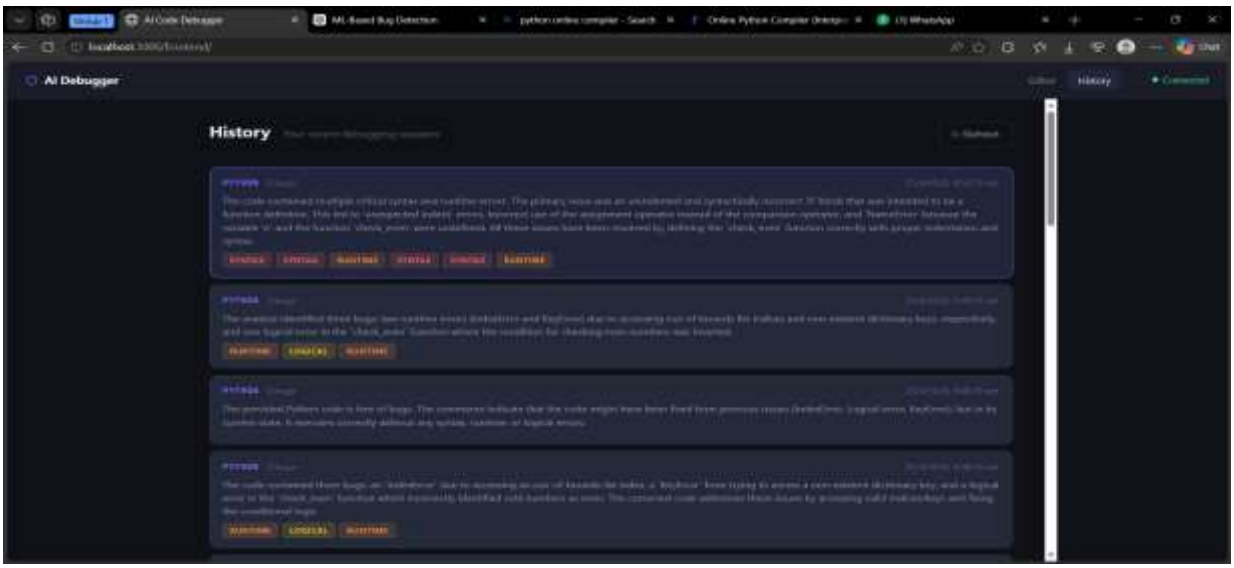

Fig. 6 History Page

This Fig.6 shows the history module stores previous code analysis results for future reference. It helps users track past bugs, fixes, and improvements over time.

VI. CONCLUSION AND FUTURE ENHANCEMENT

In conclusion, the Intelligent ML Model for Automated Bug Detection and Auto Fix Generation offers an efficient and modern solution for improving software development. By leveraging machine learning and advanced language models, the system significantly reduces manual effort, minimizes debugging time, and enhances overall code quality. It ensures accurate bug detection and provides reliable fix suggestions, making the development process faster and more efficient.

| ISSN: 2455-1864 | www.ijrai.org | editor@ijrai.org | A Bimonthly, Scholarly and Peer-Reviewed Journal |

|| Volume 9, Issue 3, May-June 2026||

DOI:10.15662/IJRAI.2026.0903008

Additionally, the system's ability to learn from past data allows it to continuously improve its performance and adapt to different coding environments. Its scalable architecture and user-friendly interface make it suitable for real-world applications and integration with existing development tools. Although certain challenges like data dependency and computational cost exist, the advantages of automation and early error detection outweigh these limitations. Overall, the system enhances developer productivity and plays a key role in building high-quality, reliable, and error-free software systems.

Future enhancements of the Intelligent ML Model for Automated Bug Detection and Auto Fix Generation can focus on improving accuracy, scalability, and real-time capabilities. The system can be extended to support multiple programming languages and handle more complex code structures. Incorporating advanced deep learning models and larger datasets can further enhance bug detection and fix generation performance. Integration with popular development environments and version control systems can make the tool more practical for developers. Additionally, optimizing computational efficiency and reducing processing time will improve overall system performance. Future work can also include enhanced security features and better handling of edge cases, making the system more robust and reliable for real world application. Continuous learning mechanisms can also be introduced, allowing the model to improve over time based on new data and user feedback. Overall, these enhancements will transform the system into a more powerful, efficient, and developer-friendly solution, making it highly suitable for real-world software development and maintenance.

REFERENCES

1. C. D. R. C. Tej, "Automated Software Bug Detection Using Machine Learning,"International Journal of Novel Research and Development (IJNRD"), ISSN: 2456-4184 vol. 10, pg no. 3, 2025.
2. B. Imran, S. Riadi, and E. Suryadi, "A Machine Learning Model for Automatic Bug Detection in Python Based on Syntactic Analysis," Jurnal Informatika, ISSN: 1978-0524 vol. 11, pg no. 2,2025.
3. J. Sun and F. Li, "Empirical Evaluation of Large Language Models in Automated Program Repair," Journal of Software Engineering Research, vol. 10, pg no. 3,2025.
4. I. Bouzenia and P. Devanbu, "An Autonomous LLM-Based Agent for Program Repair," arXiv preprint ,ISSN: 0740-7459,vol10, , pg no. 3 2024.
5. Saeed, "Automated Bug Detection with Machine Learning in Software QA," IJNRD, ISSN: 2169-3536,vol. 5, pgno. 2, Nov. 2025.
6. H. Zhang, Y. Li, and X. Chen, "Deep Learning-Based Automated Bug Detection in Software Systems," in Proc. IEEE Int. Conf. on Software Engineering (ICSE), ISSN: 0098-5589 ,vol. 5, pg no.7, 2024,
7. R. Kumar and P. Singh, "AI-Based Bug Prediction and Fix Recommendation System," in Proc. IEEE Int. Conf. on Advances in Computing, 2025, pp, ISSN: 2320-088X 45–52.
8. L. Wang, T. Liu, and Z. Zhao, "Transformer-Based Code Analysis for Automated Bug Detection," IEEE Access, ISSN: 1792-8036,vol. 12, pgno.6, 2024.
9. S. Gupta and A. Verma, "Machine Learning Techniques for Early Bug Detection in Software Development," in Proc. IEEE Int. Conf. on Smart Computing,vol.5,pg no.2 2024,
10. Y. Chen and M. Zhang, "Automated Program Repair Using Large Language Models," IEEE Software, vol. 42, pg no. 1, 2025.
11. D. Roy and K. Das, "Intelligent Code Analysis and Bug Fixing Using NLP Models," in Proc. IEEE Int. Conf. on Artificial Intelligence,vol.2, pg no.4, 2024,
12. P. Sharma and R. Mehta, "A Survey on AI-Based Bug Detection and Repair Techniques," IEEE Access, vol. 13, pg no.2 2025.
13. M. Ali, S. Khan, and N. Ahmad, "Context-Aware Automated Bug Fix Generation Using Deep Learning," in Proc. IEEE Int. Conf. on Data Science, vol. 11, pg no. 2,2025.
14. J. Lee and H. Park, "Neural Code Models for Software Defect Prediction and Repair," IEEE Transactions on Software Engineering, early access, vol. 10, pg no. 4,2025. 2025.
15. K. Patel and V. Rao, "LLM-Based Intelligent Debugging System for Modern Software Applications," in Proc. IEEE Int. Conf. on Computing and Informatics, vol.3,pg no.6 2025,
16. Emeto et al., "A Web-Based Automated Bug Detection and Fix Suggestion System," Int. J. Computer Science and Mobile Computing, vol. 14, no. 3, pg no.5,2025
17. S. Talebi et al., "Machine Learning-Driven Software Testing: Towards Autonomous Bug Detection," IARJMD, vol. 5, no. 2, pg no.1,2024

| ISSN: 2455-1864 | www.ijrai.org | editor@ijrai.org | A Bimonthly, Scholarly and Peer-Reviewed Journal |

|| Volume 9, Issue 3, May-June 2026||

DOI:10.15662/IJRAI.2026.0903008

18. D. Chhabra, "Automatic Bug Triaging Using Machine Learning-Based LLM Approach," Engineering, Technology & Applied Science Research, vol. 14, pg no.8,2024.
19. Phang and Y. Lee, "Automated Code Review and Bug Detection Using Machine Learning," in Proc. vol. 12, pgno.6, 2024.
20. "Automated Bug Detection and Correction in Software Development Using Machine Learning," Int. J. Advanced Computer Theory and Engineering, vol. 5, pgno.6, 2025.
21. N. S. Harzevili et al., "Checker Bug Detection and Repair in Deep Learning Libraries," arXiv, vol. 12, pgno.5, 2024.
22. Y. Chen et al., "A Study of Bug-Fix Patterns in Autonomous Systems," arXiv, vol. 5, pgno.6, 2025.
23. Z. Li et al., "Deep Learning for Software Bug Detection," IEEE Trans. Software Engineering, vol. 3, pgno.3, 2025.
24. S. Kim et al., "Automatic Bug Detection Using Machine Learning," in Proc. IEEE ICSE, vol. 12, pgno.6, 2024.
25. R. Manke et al., "Bug Localization in Deep Learning Programs," arXiv, vol. 1, pgno.8, 2024.
26. Seedha Devi, V., Nivedha, S., Harisha, V., Mol, D. R., & Janaranjini, J. R. (2026). Enhanced prediction of PCOS and PCOD using deep learning for early diagnosis and clinical risk stratification. International Journal of Advanced Research in Computer Science & Technology (IJARCST), 9(3), 783–793.
27. Seedha Devi, V., Kumar, M. D., & Kumar, C. A. (2026). Flutter-based SOS alert and location tracking application with volunteer assist and rescue. International Journal of Research and Applied Innovations (IJRAI), 9(3), 521–530. https://doi.org/10.15662/IJRAI.2026.0903003
28. Seedha Devi, V., Selvi, D., Uma Maheshwari, K., & Yuvashree, G. (2026). Food linker: A smart system for global waste reduction. International Journal of Engineering & Extended Technologies Research (IJEETR), 8(3), 5012–5021. https://doi.org/10.15662/IJEETR.2026.0803002
29. Seedha Devi, V., Namitha, B., Divya Dharshini, J., & Livetha, K. (2026). A hybrid biometric and geo-fencing based smart attendance system. International Journal of Advanced Research in Computer Science and Technology (IJARCST), 9(3), 794–802. https://doi.org/10.15662/IJARCST.2026.0903002
30. Alangaram, S., Praveen, S., Rajesh, V., & Sanjai, A. (2026). Sales guard AI-driven decision intelligence platform for business optimization. International Journal of Engineering & Extended Technologies Research (IJEETR), 8(3), 5022–5031. https://doi.org/10.15662/IJEETR.2026.0803003
31. Seedha Devi, V., Harshini, R., Dhana Lakshmi, E., Gayathri, N., & Nithesha, P. (2026). Low-code mobile application builder with AI-assisted features using Flutter & Firebase. International Journal of Research Publications in Engineering, Technology and Management (IJRPETM), 9(3), 1001–1010.
32. Seedha Devi, V., Harshini, R., Dhana Lakshmi, E., Gayathri, N., & Nithesha, P. (2026). Low-code mobile application builder with AI-assisted features using Flutter & Firebase. International Journal of Research Publications in Engineering, Technology and Management (IJRPETM), 9(3), 1001–1010. https://doi.org/10.15662/IJRPETM.2026.0903001
33. Seedha Devi, V., Divya Narasimman, S., Jayaprakash, S., & Mohamed Suhel, H. N. (2026). Smart IoT-based pedestrian power generator using DC motor. International Journal of Computer Technology and Electronics Communication (IJCTEC), 9(3), 990–999. https://doi.org/10.15680/IJCTECE.2026.0903002
34. Seedha Devi, V., Mahalakshimi, P. V., & Anitha, A. (2026). Automated skin disease analysis and detection using AI-powered mobile application. International Journal of Research and Applied Innovations (IJRAI), 9(3), 531–539. https://doi.org/10.15662/IJRAI.2026.0903004
35. Alangaram, S., Udaykiran, M., Rajkumar, K., & Yogeeswaran, T. (2026). Enhancing customer churn prediction and retention for e-commerce. International Journal of Advanced Research in Computer Science & Technology (IJARCST), 9(3), 803–813. https://doi.org/10.15662/IJARCST.2026.0903003
36. Alangaram, S., Kiswar, M., Ajay, B., & Ezhilkumaran, P. (2026). Socialflow AI: Voice to social media scheduler. International Journal of Research and Applied Innovations (IJRAI), 9(3), 540–547. https://doi.org/10.15662/IJRAI.2026.0903005
37. Raghul, K., Rajasolan, P., Rohinth, S., & Tharun, P. (2026). AI knowledge sharing web portal. International Journal of Advanced Research in Computer Science & Technology (IJARCST), 9(3), 814–823. https://doi.org/10.15662/IJARCST.2026.0903004
38. Sangeetha, D., Dharan, K. D., Krishna, A. C., & Karthikeyan, C. (2026). Speech and text conversion system for sign language using ML. International Journal of Computer Technology and Electronics Communication (IJCTEC), 9(3), 1000–1007. https://doi.org/10.15680/IJCTECE.2026.0903003
39. Seedha Devi, V., Kaavya, S., Deepika, B., Jayashree, D., & Nithikaa, L. (2026). AI-driven voter authentication and fraud detection system. International Journal of Computer Technology and Electronics Communication (IJCTEC), 9(3), 1008–1017. https://doi.org/10.15680/IJCTECE.2026.0903004

| ISSN: 2455-1864 | www.ijrai.org | editor@ijrai.org | A Bimonthly, Scholarly and Peer-Reviewed Journal |

|| Volume 9, Issue 3, May-June 2026||

DOI:10.15662/IJRAI.2026.0903008

40. Alangaram, S., Yuvaraj, G., Srivatsan, M. J., & Sathish, R. (2026). An IoT-based smart helmet for real-time rider safety monitoring and emergency response system. International Journal of Research in Production Engineering, Technology and Management (IJRPETM), 9(3), 1021–1030. https://doi.org/10.15662/IJRPETM.2026.0903003
41. Raghul, K., Thamaraikannan, R., Sunil Kumar, S., & Siva, B. (2026). Plastitrack: A community-driven plastic waste collection and redemption platform. International Journal of Research and Applied Innovations (IJRAI), 9(3), 548–557. https://doi.org/10.15662/IJRAI.2026.0903006
42. Dr. V. Seedha Devi, M. Parvinraj, J. Dinesh, M. Venkatramana, & P. Suryaprakash Raj. (2026). Darkshield: Mobile intrusion detection using post-authentication failure analysis and Android security APIs. International Journal of Advanced Research in Computer Science & Technology (IJARCST), 9(3), 824–833. https://doi.org/10.15662/IJARCST.2026.0903005
43. S. Alangaram, K. Mugunthan, R. Elango, & J. J. Harish. (2026). AI powered secure payment with eye recognition in wallet platform. International Journal of Computer Technology and Electronics Communication (IJCTEC), 9(3), 1018–1025. https://doi.org/10.15680/IJCTECE.2026.0903005
44. Dr. V. Seedha Devi, Priyadharshini R., Vaishnavi P. S., & Shalini M. (2026). Cybersecurity enhancement in electric vehicle systems using principal component analysis (PCA). International Journal of Research and Applied Innovations (IJRAI), 9(3), 558–568. https://doi.org/10.15662/IJRAI.2026.0903007
45. Dr. V. Seedha Devi, D. Yogeshwari, B. Reshma, & B. Sowmiya. (2026). Ayursutra Panchakarma patient management and therapy scheduling software – AI powered chatbot assistance. International Journal of Engineering & Extended Technologies Research (IJEETR), 8(3), 5032–5041. https://doi.org/10.15662/IJEETR.2026.0803004