iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/

PROGRAM ERROR ANALYSIS SOFTWARE

Rohini thangam[1] and Dr.S.Prathi [2]
[1]UG Student, Department of Computer Applications Vels Institute of Science, Technology and Advanced Studies (VISTAS), Pallavaram, Chennai – 600 117, India
[2]Assistant Professor Department of Computer Applications Vels Institute of Science, Technology and Advanced Studies (VISTAS), Pallavaram, Chennai – 600 117, India

ABSTRACT:
One of the most significant barriers to mastering programming is the frustration caused by cryptic, machine-oriented error messages. For many students, these technical "walls of text" offer no guidance, often turning a simple mistake into a major roadblock. ErrExplain is developed to solve this problem by transforming the debugging process into a conversational, learning-centred experience. Built as a full-stack platform using the Flask framework and a SQLite database, ErrExplain provides a workspace that speaks a human language. Instead of leaving users to guess why their code failed, the system's custom analysis engine breaks down errors into simple, natural explanations that help them actually understand the "why" behind their mistakes. The platform features a curated library of many practice problems, a modern "Master-Detail" interface for focused learning, and secure session-based tracking to help users monitor their growth over time. By combining these technical features with a deep empathy for the learner's journey, ErrExplain acts as a 24/7 digital mentor, empowering students to build the foundational logic and confidence they need to transition from beginners to proficient developers. Rather than acting as a simple technical parser, functions as a linguistic bridge. It captures raw, intimidating stack traces and translates them into empathetic, natural language narratives. By utilizing a Python-driven backend and a user-centric frontend, the system identifies the "root of the misunderstanding" and explains it using relatable analogies and actionable steps. The uniqueness lies in its philosophy: it treats an error not as a failure of the programmer, but as a temporary lapse in communication. By prioritizing cognitive accessibility and educational scaffolding, the project reduces the "time-to-solution" while simultaneously building the developer's confidence. Ultimately, ErrExplain transforms the loneliest part of coding, debugging into a guided, supportive, and human-centric learning process.

Keywords:
Error Explanation, Automated Debugging, User-Friendly Learning, Code Analysis, Natural Language Explanation, Python, Flask, Human Computer Interaction

1.INTRODUCTION

Programming is often celebrated as an act of pure logic, but anyone who has spent a long night staring at a blinking cursor knows that it is, in reality, a deeply emotional experience. It is a process of trial, error, and all too often profound isolation. We are taught to write in languages that feel like poetry, yet when we make a mistake, the machine responds with a cold, mechanical silence or a wall of red text that feels more like a reprimand than a guide.

ErrExplain was born out of a simple observation: a mistake is not a failure; it is a conversation that hasn't finished yet. The vision for this project was to move past the rigid "Correction-Response" loop of traditional compilers and move toward a "Mentorship" model. We wanted to build a tool that looks at a broken line of code and sees the human effort behind it.

If a student tries to access a list item that doesn't exist, they don't need to be told they've committed a "Logical Violation." They need a friend to say, "It looks like you're reaching for something that hasn't been put on the shelf yet." This shift—from technical jargon to empathetic analogy is the core of ErrExplain.

The technical foundation of this project (built on a Python-driven full-stack architecture) serves a higher purpose: preserving the dignity and motivation of the developer. By intercepting cryptic error logs and translating them into natural, conversational narratives, we are doing more than just debugging code. We are lowering the barrier to entry for the next generation of creators.

We believe that the future of technology shouldn't just be about making machines faster; it should be about making them more accessible, more understanding, and ultimately, more human. ErrExplain is our contribution to that

iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/

future—a tool that ensures that when a developer reaches their breaking point, they find a solution instead of a wall.

2.RELATED WORK

Traditional debugging systems generate technical error messages that are often difficult for beginners to understand. Foundational work by Alfred V Aho et al. focuses on error detection through compilers, but these approaches prioritize machine-level accuracy over user comprehension. Similarly, software engineering principles by Ian Sommerville and Roger S Pressman emphasize system design and maintainability but do not address user-friendly error interpretation.

Recent research in Human Computer Interaction highlights the importance of usability and learning support. Concepts like Cognitive Load Theory by John Sweller suggest simplifying complex information for better understanding. However, existing tools still lack a human-centred explanation layer. ErrExplain addresses this gap by converting technical errors into simple, meaningful explanations, improving both debugging efficiency and learning.

3.HARDWARE AND SOFTWARE REQUIREMENTS

3.1 Hardware Requirements

- Processor: Multi-core CPU (i5/Ryzen 5 or higher)
- Memory (RAM): Minimum 8GB (16GB recommended)
- Storage: 256GB SSD (fast storage preferred)
- Display: Full HD (1920×1080 resolution)

3.2 Software Requirements

- Operating System: Windows 11 / macOS / Ubuntu (cross-platform support)
- Primary IDE: Visual Studio Code (extensions + terminal + Git support)
- Version Control: Git & GitHub (code management & collaboration)
- Web Browser: Google Chrome / Brave (powerful developer tools)
- Framework: Flask
- Programming language: Python 3.10+
- Frontend: html, CSS, react.js, tailwind CSS, JavaScript

4.WORKING METHODOLOGY

The ErrExplain system follows a step-by-step methodology to convert complex errors into simple explanations:

1. **Error Capture**
   The system captures runtime errors using Python's traceback mechanism and converts them into a structured JSON format.
2. **Data Cleaning**
   Unnecessary details like memory addresses, file paths, and internal library calls are removed using Regular Expressions (Regex).
3. **Error Identification**
   The cleaned error is analysed to detect the exact error type (e.g., syntax error, key error).
4. **Knowledge Mapping**
   The identified error is matched with a predefined knowledge base to fetch a simple explanation and real-world analogy.
5. **Solution Generation**
   The system generates a clear explanation along with step-by-step fixes to resolve the issue.
6. **User Display**
   The processed output is sent to the frontend and displayed dynamically in an easy-to-understand format

5.RESULTS AND DISCUSSION

The implementation of ErrExplain demonstrates strong and consistent results by effectively simplifying complex system-generated error messages into clear, user-friendly explanations. This significantly reduces the cognitive load on developers, especially beginners, and minimizes the time spent on debugging. The system not only identifies the root cause of errors but also provides structured solutions and guidance, improving overall

iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/

development efficiency. The backend, powered by Python and Flask, ensures reliable processing of incoming error data, while the frontend built with React offers a smooth and responsive user experience. The integration of regex-based data cleaning and dictionary-based mapping enhances both the accuracy and speed of the system. The system performs well under normal conditions and maintains quick response times due to optimization techniques like precompiled regex patterns. Additionally, the inclusion of a structured knowledge base allows the system to deliver meaningful explanations rather than generic outputs. However, the system still faces challenges when dealing with highly complex, chained, or uncommon errors, where deeper contextual understanding may be required. Performance may also slightly degrade with extremely large logs or high-frequency requests.

Despite these limitations, the overall system proves to be robust, efficient, and highly practical. It bridges the gap between machine-level error outputs and human understanding, making debugging not only faster but also more educational. The approach highlights how combining intelligent backend processing with a user-centric frontend design can significantly improve the software development experience.

6.APPLICATIONS

ErrExplain has wide applications in software development, where it helps developers quickly understand and resolve errors by converting complex technical messages into simple explanations.

This reduces debugging time and improves productivity. It is also highly useful in educational environments, enabling beginners to learn programming more effectively by clearly explaining mistakes and providing step-by-step solutions. The system can be integrated into development tools like Visual Studio Code to offer real-time error assistance during coding.

In addition, ErrExplain is valuable in web development using technologies such as React and Flask, where frequent debugging is required. It also supports testing and quality assurance by helping testers identify and interpret issues quickly.

Furthermore, technical support teams can use it to explain errors to non-technical users in a simple and understandable way. Overall, the system enhances efficiency, learning, and communication across multiple domains.

7.CONCLUSION

The ErrExplain system successfully demonstrates how complex technical error messages can be transformed into simple, human-readable explanations. By combining a powerful backend using Python and Flask with an interactive frontend built on React, the system provides a fast, efficient, and user-friendly debugging solution. It reduces the time and effort required to identify and fix errors, making the development process smoother and more productive.

Furthermore, the system highlights the importance of clean architecture, optimized data processing, and user-centric design. Although there are minor limitations such as dependency on the knowledge base and performance with very large logs, the overall implementation proves to be reliable and scalable. In conclusion, ErrExplain serves as an effective tool that bridges the gap between complex system errors and user understanding, making debugging easier, faster, and more accessible for developers of all skill levels.

8. ACKNOWLEDGEMENTS

The authors gratefully acknowledge the Department of Computer Applications, Vels Institute of Science, Technology and Advanced Studies (VISTAS), Pallavaram, Chennai, for providing the laboratory infrastructure and technical support required to complete this research work.

REFERENCES

1) I. Sommerville, Software Engineering, 10th ed., Pearson, 2016.
2) R. S. Pressman and B. R. Maxim, Software Engineering: A Practitioner's Approach, 8th ed., McGraw-Hill, 2019.
3) R. C. Martin, Clean Code: A Handbook of Agile Software Craftsmanship, Prentice Hall, 2008.
4) E. Gamma, R. Helm, R. Johnson, and J. Vlissides, Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley, 1994.
5) A. V. Aho, M. S. Lam, R. Sethi, and J. D. Ullman, Compilers: Principles, Techniques, and Tools, 2nd ed., Pearson, 2006.

iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/

6) [A. Silberschatz, H. F. Korth, and S. Sudarshan, Database System Concepts, 7th ed., McGraw-Hill, 2019.
7) A. S. Tanenbaum and H. Bos, Modern Operating Systems, 4th ed., Pearson, 2015.
8) J. Sweller, "Cognitive load during problem solving: Effects on learning," Cognitive Science, vol. 12, no. 2, pp. 257–285, 1988.
9) J. Piaget, The Psychology of Intelligence, Routledge, 1950.
10) [S. K. Card, T. P. Moran, and A. Newell, The Psychology of Human-Computer Interaction, CRC Press, 1983.
11) B. Schneiderman et al., Designing the User Interface: Strategies for Effective Human-Computer Interaction, 6th ed., Pearson, 2016.
12) M. Fowler, Refactoring: Improving the Design of Existing Code, 2nd ed., Addison-Wesley, 2018.