Google Summer of Code

GSOC 2026

PROJECT TITLE - Advanced Fuzzing and Image Injection Framework for EROFS Kernel and erofs-utils

ORGANIZATION – EROFS filesystem

By – **Daksh Garg**

GSOC 2026 PROPOSAL FOR EROFS FILESYSTEM

ABOUT ME

NAME – DAKSH GARG

TIMEZONE – **Asia/Kolkata (UTC +05:30)**

COUNTRY – INDIA

PHONE - +91 8076934841

EMAIL – dgav3105@gmail.com

GITHUB – https://github.com/thedgarg31

LEETCODE – https://leetcode.com/u/thedgarg31/

LINK TO RESUME -
https://drive.google.com/file/d/1LxG0_xFO8srcB8awvTCcpN9n_jYRM32j/view?usp=sharing

I'm a **passionate and dedicated developer** driven by curiosity and a constant thirst for learning. I strongly believe in the power of process over mere results, following the principle of *"strive for competence, success will follow."* With a **logical, focused, and hardworking mindset**, I continuously seek opportunities to grow, contribute, and refine my skills.

I am currently pursuing a **B.Tech in Computer Science and Engineering (Data Science)** from Amity University, Noida, India, along with a **BS in Data Science from IIT Madras, India**.

Beyond coding, I enjoy **playing chess, guitar, and reading books**, which help me stay creative and maintain a balanced perspective.

I have actively contributed to **open source** through programs like **JWOC**, where I improved **SEO performance, enhanced website accessibility, and optimized code efficiency** across projects. My contributions span multiple repositories including:

- **Kornia-rs** – implementing optical flow algorithms
- **FIMS** – expanding test suites
- **Joomla** – enhancing accessibility features
- **name3ss** – identifying and working on real-world issues, Contribution link: https://github.com/namel3ss-Ai/namel3ss/issues/13

These experiences demonstrate my ability to **work on real-world problem statements, collaborate in distributed teams, and deliver meaningful improvements** in open-source environments.

My technical journey includes **Data Structures and Algorithms in C++ (Coding Ninjas)**, **AR/VR (IIT Madras & Finland University collaboration)**, and hands-on experience in **AI,**

ML, web development, Rust, and biomedical informatics. I have also worked extensively on **frontend optimizations, responsiveness fixes, and UI/UX improvements** across various repositories.

Beyond open source, I am building innovative projects such as:

- **AI-powered Legal Document Analyzer** – simplifying legal jargon and highlighting risky clauses using Python, GPT-4 API, and LangChain
- **Smart Community Health Monitoring System (SCHMS)** – an IoT-enabled platform leveraging AI/ML and predictive analytics
- **AR/VR-based Library** – creating an immersive and interactive reading environment
- **Mental Health Sentiment Analysis** – applying NLP and deep learning models to analyze trends

Additionally, I participated in the **Apple Student Swift Challenge 2025**, demonstrating my skills in **Swift-based application development**.

I have a strong foundation in **Python, C++, Java, Rust, R, Swift, React.js, Flask, Django, Node.js, MongoDB, and IoT platforms**.

Through **GSoC**, I aim to **deepen my technical expertise, collaborate with experienced mentors, and contribute meaningfully to impactful open-source projects**. I see this as an opportunity to strengthen my **problem-solving abilities, engage with a global developer community, and build long-term contributions**.

In the long run, I aspire to work on **innovative technology solutions that create real-world impact**, while continuously learning and evolving as a developer.

PRIOR DEVELOPMENT EXPERIENCE

I have been actively contributing to open-source projects on GitHub, where I've worked across backend, frontend, and testing-related tasks. Some of my key contributions include:

• **Kornia-rs** – Worked on implementing the Pyramidal Lucas-Kanade Optical Flow algorithm, improving motion tracking accuracy.

• **FIMS** – Contributed by adding a C++ testing framework and cleaning up unnecessary files in the repository to improve overall structure and maintainability.

• **Unify** – Fixed multiple UI/UX issues and improved frontend responsiveness, making the interface more accessible and user-friendly.

• **FIMS GitHub Action Proposal** – Proposed an automated documentation generation workflow in the CI/CD pipeline to simplify and streamline the development process.

• **JWOC Contributions** – Worked on resolving SEO-related issues and improving website responsiveness across different screen sizes.

• **Personal Work & Open Source** – Built interactive web applications and contributed to various open-source projects in **Rust and Python**, focusing on performance improvements and usability.

Through these experiences, I've developed a strong understanding of **collaborative development, clean coding practices, and working with real-world codebases**.

TIME COMMITMENT

I am committed to dedicating a minimum of 20 hours per week to this project. I understand the importance of timely progress and am flexible to adjust my schedule to accommodate project needs, potentially exceeding the initial commitment as required.

WHY I CHOOSE THIS ORGANIZATION AND PROJECT

The EROFS project genuinely stood out to me because of its strong real-world impact in systems that people actually use at scale — from Android devices to container infrastructures. What I found especially interesting is how EROFS is not just another filesystem, but a carefully designed, high-performance solution focused on reliability, efficiency, and modern use cases like immutable infrastructure and container images.

The Advanced Fuzzing and Image Injection project immediately caught my attention because it works at the core of system reliability and security. Since EROFS is designed to handle data from potentially untrusted sources, ensuring its robustness is critical. The idea of building tools that can actively detect bugs, uncover edge cases, and strengthen the filesystem against unexpected inputs is something I find both challenging and meaningful.

Honestly, this project just clicked with me. It's not just about writing code — it's about making systems safer and more reliable. I've always been interested in understanding how systems behave under stress, and the opportunity to design fuzzing frameworks and generate adversarial test cases feels like the kind of problem-solving I genuinely enjoy.

What excites me the most is the combination of low-level system understanding and practical impact. Working on fuzzing tools, testing pipelines, and image injection mechanisms will allow me to explore how filesystems behave internally, while also contributing directly to improving the stability of a widely used Linux component. It's the kind of work where improvements are not just theoretical — they directly prevent crashes and improve real-world reliability.

At the same time, this project offers a strong learning opportunity. It will help me deepen my understanding of filesystem internals, kernel interactions, and advanced testing techniques like fuzzing and automated validation. I'm also excited about the chance to work with experienced maintainers in the Linux ecosystem and learn how large-scale, performance-critical systems are developed and maintained.

Overall, I chose this organization and project because it aligns perfectly with my interest in system-level programming, testing, and building reliable software. It gives me the opportunity to contribute to something widely impactful while continuously learning and improving as a developer.

PROJECT OVERVIEW

EROFS (Enhanced Read-Only File System) is a high-performance, immutable filesystem widely used in Linux environments, including Android systems and container images. Its design focuses on efficiency, compression, and reliability in read-only scenarios.

However, as EROFS continues to evolve, ensuring robustness against malformed inputs, corrupted images, and edge cases becomes critical. Since EROFS is designed to handle data from potentially untrusted sources (e.g., container images), maintaining correctness and security is essential.

This project focuses on designing and implementing an advanced fuzzing and image injection framework to test and strengthen both the EROFS kernel implementation and erofs-utils. The goal is to proactively detect bugs, improve resilience, and enhance the reliability of the filesystem under real-world and adversarial conditions.

PROBLEM STATEMENT

Although EROFS has a robust design, there are several challenges in maintaining its reliability and security:

- **Limited Fuzzing Coverage:**
Existing fuzzing tools are present but not comprehensive enough to cover diverse filesystem states and edge cases.

- **Lack of Advanced Image Injection Testing:**
There is no structured mechanism to generate and test malformed or adversarial filesystem images systematically.

- **Increasing Complexity of Features:**
Features such as compression, deduplication, and layering increase the surface area for potential bugs.

- **Kernel-Level Risk:**
Since EROFS operates within the Linux kernel, any bug can lead to crashes or vulnerabilities when handling untrusted images.

- **Insufficient CI Integration:**
Continuous fuzzing and automated testing are not fully integrated into CI workflows for ongoing validation.

PROPOSED SOLUTION

To address these challenges, I propose building a comprehensive fuzzing and testing framework with the following components:

1. **Advanced Fuzzing Engine:**
   Develop a modular fuzzing system using tools like AFL++ or libFuzzer to test EROFS parsing logic.
2. **Image Injection Tool:**
   Create a tool to generate malformed, edge-case, and adversarial filesystem images for testing robustness.
3. **Integration with erofs-utils and Kernel Testing:**
   Apply fuzzing to both userspace tools and kernel-level implementations.
4. **CI/CD Integration:**
   Automate fuzzing workflows using GitHub Actions to ensure continuous validation.
5. **Crash Analysis & Reporting:**
   Implement logging and reporting mechanisms for identifying and debugging failures efficiently.

The system will be designed to be scalable, maintainable, and extensible for future testing needs.

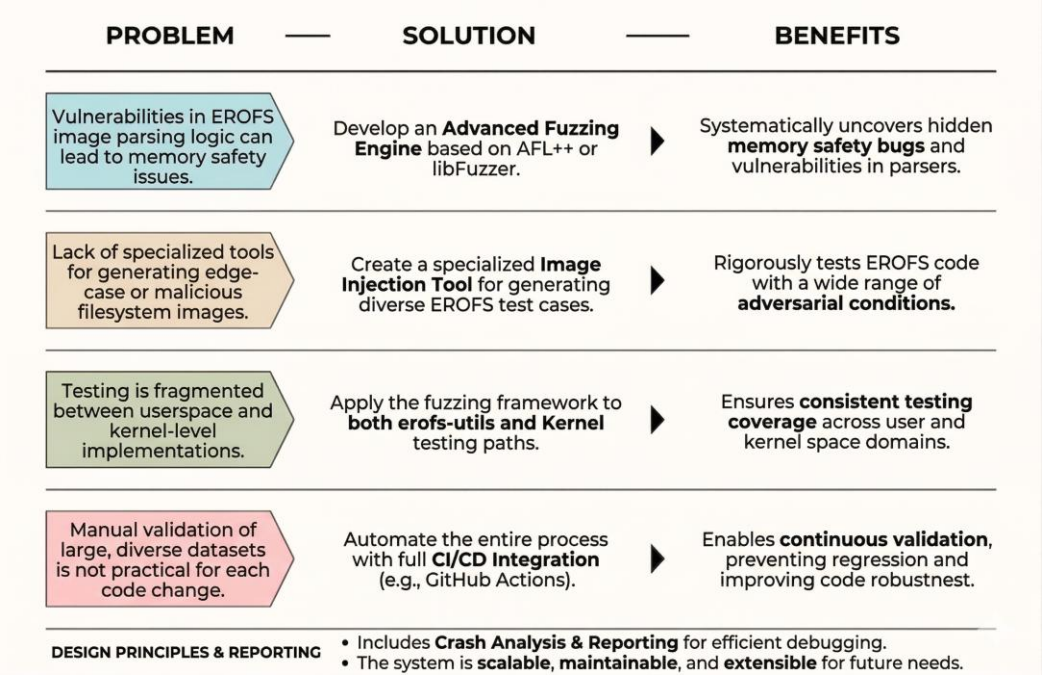

Figure 1: Understanding of problem-solution-benefit relationship of the project *Source: Author*

PROJECT TIMELINE & DELIVERABLES

PHASE 1: WEEKS 1–4 – RESEARCH, SETUP & INITIAL FRAMEWORK

Objectives:

- **Understanding EROFS Architecture and Existing Tools:**
  - Study the EROFS filesystem design, focusing on on-disk format, compression, and metadata handling.
  - Explore erofs-utils (especially fsck.erofs) to understand current validation and unpacking workflows.
  - Analyze existing fuzzing approaches and identify gaps in coverage.
- **Environment Setup and Tooling:**
  - Set up development environment for erofs-utils and kernel testing.
  - Integrate basic fuzzing tools such as AFL++ or libFuzzer.
  - Prepare sample EROFS images for initial testing and benchmarking.
- **Initial Fuzzing Prototype:**
  - Develop a basic fuzzing setup targeting critical input parsing components.
  - Run initial fuzz tests to identify obvious edge cases or crash scenarios.
  - Log and analyze early results to refine fuzzing strategy.
- **Design of Image Injection Strategy:**
  - Define how malformed and edge-case filesystem images will be generated.
  - Identify key structures (superblock, inodes, compressed clusters) for targeted mutation.

PHASE 2: WEEKS 5–8 – CORE DEVELOPMENT & INTEGRATION

Objectives:

- **Advanced Fuzzing Engine Development:**
  - Extend the fuzzing framework to cover multiple components of erofs-utils.
  - Support structured fuzzing for filesystem metadata instead of purely random input.
  - Improve coverage by targeting compression, deduplication, and metadata parsing paths.
- **Image Injection Tool Implementation:**
  - Build a tool to generate malformed and adversarial EROFS images.
  - Introduce controlled mutations in filesystem structures to simulate real-world corruption scenarios.
  - Ensure generated images are diverse and useful for testing different failure modes.
- **Kernel-Level Testing Integration:**
  - Extend testing to kernel-level EROFS handling where feasible.
  - Identify crash points, memory issues, or unexpected behavior during mounting or parsing.

- **Automation and CI/CD Integration:**
  - Integrate fuzzing workflows into GitHub Actions or similar CI pipelines.
  - Enable periodic automated fuzzing runs with logging and reporting.
- **Intermediate Testing and Feedback:**
  - Share results and progress with mentors.
  - Refine approach based on feedback and observed limitations.

PHASE 3: WEEKS 9–12 – OPTIMIZATION, ANALYSIS & FINALIZATION

Objectives:

- **Performance Optimization of Fuzzing Framework:**
  - Improve execution speed and reduce overhead of fuzzing runs.
  - Optimize mutation strategies to maximize meaningful test coverage.
  - Balance performance with memory usage and stability.
- **Crash Analysis and Debugging Framework:**
  - Implement structured logging for crashes and unexpected behaviors.
  - Categorize bugs based on severity and reproducibility.
  - Create reproducible test cases for identified issues.
- **Comprehensive Testing and Validation:**
  - Perform large-scale fuzzing runs across different configurations.
  - Validate stability improvements and ensure no regressions are introduced.
  - Test across multiple environments where possible.
- **Documentation and Developer Guide:**
  - Document the fuzzing framework, usage instructions, and design decisions.
  - Provide guidelines for extending fuzzing coverage in the future.
  - Include examples of detected issues and how they were resolved.
- **Final Deliverables and Submission:**
  - Deliver a complete fuzzing and image injection framework.
  - Provide CI-integrated workflows for continuous validation.
  - Submit final reports, documentation, and code for review.

FINAL DELIVERABLES

- Advanced fuzzing framework for erofs-utils and EROFS components
- Image injection tool for generating adversarial filesystem images
- CI-integrated automated fuzzing pipeline
- Crash analysis and reporting system
- Documentation and developer guidelines

This timeline ensures steady progress from understanding the system to building a robust testing framework, with sufficient time for optimization and real-world validation.

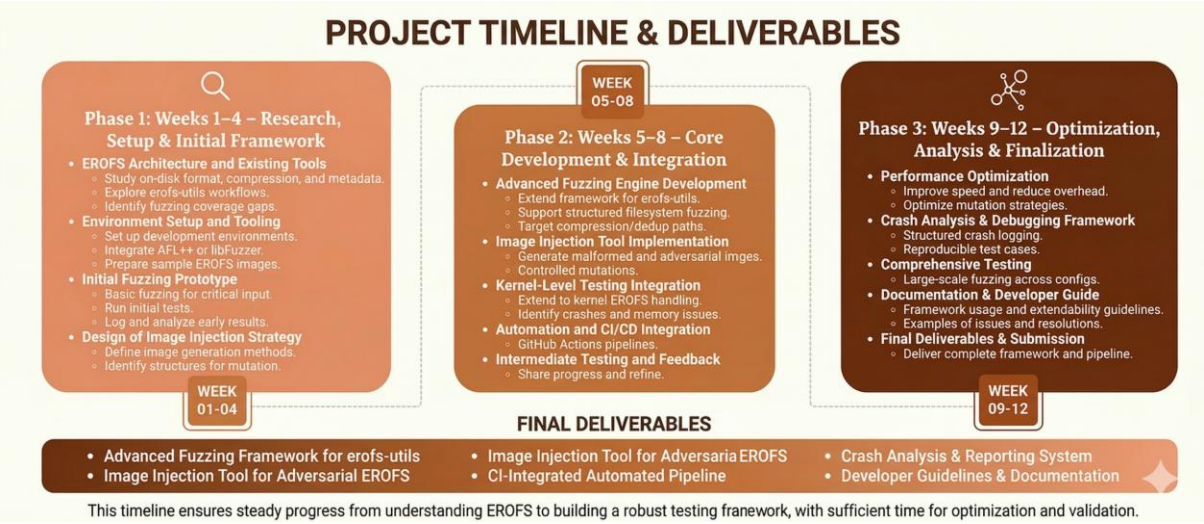

Figure 2: Project timeline and milestones *Source: Author*

TECHNOLOGIES & TOOLS

• **Languages:** C (primary), Python (for scripting and automation), Rust/Go (optional for tooling)

• **Fuzzing Frameworks:** AFL++, libFuzzer

• **Testing & Debugging Tools:** GDB, AddressSanitizer (ASAN), UndefinedBehaviorSanitizer (UBSAN), Valgrind

• **Filesystem Tools:** erofs-utils (fsck.erofs, mkfs.erofs), Linux kernel testing environment

• **CI/CD & Automation:** GitHub Actions

• **Build Tools:** GCC/Clang, Make

• **Version Control:** Git & GitHub

KEY BENEFITS & IMPACT

1. **Improved Filesystem Reliability:**
   • Detects hidden bugs and edge cases in EROFS parsing and handling.
   • Strengthens stability when dealing with real-world and malformed inputs.

2. **Enhanced Security:**
   • Identifies vulnerabilities caused by corrupted or adversarial filesystem images.
   • Reduces risk of kernel crashes when handling untrusted data.
3. **Comprehensive Testing Coverage:**
   • Expands testing across multiple filesystem components including metadata, compression, and deduplication.
   • Improves confidence in filesystem correctness.
4. **Automated Continuous Validation:**
   • Integrates fuzzing workflows into CI pipelines for continuous testing.
   • Ensures long-term stability as new features are added.
5. **Efficient Debugging and Issue Tracking:**
   • Structured crash logging and reproducible test cases improve debugging efficiency.
   • Helps developers quickly identify and fix issues.
6. **Improved Developer Experience:**
   • Provides tools and documentation for future contributors to extend testing easily.
   • Encourages better testing practices within the community.
7. **Real-World Impact:**
   • Improves reliability of systems using EROFS (Android, containers, cloud infrastructure).
   • Contributes to safer and more robust Linux ecosystem components.

STRETCH GOALS (BEYOND THE PROPOSED SCOPE)

In addition to the core deliverables, I plan to:

• **Coverage-Guided Smart Fuzzing:**
Develop smarter mutation strategies based on code coverage to improve efficiency of fuzzing.

• **Differential Testing Framework:**
Compare EROFS behavior across different configurations or tools to detect inconsistencies.

• **Visualization Dashboard for Fuzzing Results:**
Create a simple dashboard to track crashes, coverage, and test progress over time.

• **Extended Kernel Fuzzing Support:**
Expand fuzzing coverage deeper into kernel-level EROFS handling where feasible.

• **Performance Benchmarking for Fuzzing:**
Measure fuzzing efficiency and optimize execution speed and resource usage.

POST GSOC

My Commitment to Long-Term Contribution

Through this project, I aim to gain a deeper understanding of filesystem internals, testing methodologies, and kernel-level reliability. My goal is not limited to contributing during GSoC, but to continue as an active contributor to the EROFS ecosystem.

With this mindset, I would like to propose the following long-term improvements:

1. **Continuous Fuzzing Integration with CI/CD**
   Implementation:
   • Extend CI pipelines to run periodic fuzzing jobs.
   • Automate reporting and issue tracking for detected bugs.

Benefits:
• Ensures continuous reliability testing
• Detects regressions early

---

2. **Advanced Crash Analysis Framework**
   Implementation:
   • Build tools to automatically classify and prioritize crashes.
   • Generate reproducible test cases for each issue.

Benefits:
• Faster debugging and resolution
• Better maintainability

---

3. **Interactive Testing and Debugging Tools**
   Implementation:
   • Provide utilities for developers to easily test custom EROFS images.
   • Improve usability of fuzzing tools for contributors.

Benefits:
• Improves developer productivity
• Simplifies onboarding

---

4. **Documentation and Learning Resources**
   Implementation:
   • Create guides explaining fuzzing workflows and filesystem testing.
   • Add examples and case studies of detected issues.

Benefits:
• Helps new contributors understand system internals
• Encourages community participation

---

5. **Integration with Static Analysis Tools**
   Implementation:
   • Combine fuzzing with static analysis tools (e.g., Clang-Tidy).
   • Detect potential issues before runtime.

Benefits:
• Improves code quality
• Reduces bugs early in development

By working on these improvements, I aim to contribute not only a functional fuzzing system during GSoC but also a sustainable testing ecosystem that continues to benefit EROFS development in the long term.

These enhancements reflect my commitment to building reliable, maintainable, and production-quality systems while actively contributing to the open-source community.

DEEP ENGAGEMENT DURING THE COMMUNITY BONDING PERIOD

During the community bonding period, I plan to actively engage with mentors and the EROFS community to build a strong foundation before development begins:

• **Guidance on Navigating the Codebase:**
  • I will explore the erofs-utils and relevant kernel components to understand the architecture.
  • I would appreciate mentor guidance on the best approach to study critical modules and workflows.

• **Clarification on Project Scope and Priorities:**
  • I will discuss with mentors to understand immediate goals and long-term expectations.
  • This will help me align my development plan with real project needs.

• **Understanding Filesystem Internals and Data Flow:**
  • I aim to understand how EROFS handles metadata, compression, and filesystem structures.
  • This will help me design effective fuzzing strategies and image injection techniques.

• **Early Contributions and Setup:**
  • I will start with small contributions, setup improvements, or minor fixes to get familiar with the workflow.
  • This will help me adapt to coding standards and development practices.

• **Validating My Understanding with Mentors:**
  • I will share my understanding through notes or diagrams.
  • Seek feedback to ensure I am on the right track before full development begins.

MENTORSHIP & COLLABORATION

Throughout the project, I will:

• **Actively Engage with Mentors:**
  • Seek regular feedback on design decisions, implementation strategies, and progress.
  • Share updates on completed work, challenges, and next steps.

• **Participate in GitHub Discussions and PR Reviews:**
  • Actively contribute to issues, discussions, and pull requests in the EROFS repository.
  • Learn from existing PRs and community feedback.

• **Collaborate with the Open Source Community:**
  • Incorporate feedback from maintainers and contributors.
  • Participate in technical discussions to improve solutions.

• **Maintain Consistent Communication:**
  • Provide weekly progress updates.
  • Be responsive to feedback and iterate quickly.
  • Ask for guidance early when facing challenges.

Presentation and Knowledge Sharing:
  • I would be excited to present my work and outcomes to the EROFS community.
  • Sharing results will help me gain feedback and improve further.

• **Learning from Experienced Maintainers:**
  • I look forward to learning about filesystem internals, kernel-level development, and testing strategies.
  • Understanding how large-scale Linux systems are built will be highly valuable.

• **Long-Term Contribution:**
  • I intend to continue contributing to EROFS beyond GSoC by improving tools and fixing issues.
  • I would like to stay involved in system-level open-source projects.

PREFERRED COMMUNICATION METHODS

• **Primary Communication:** GitHub (issues, pull requests, discussions)
• **Secondary Communication:** Email updates with mentors
• **Virtual Check-ins:** Meetings via Zoom/Matrix/other preferred platforms

---

COMMITMENT

I am genuinely committed to contributing meaningfully throughout the GSoC period. I approach my work with consistency, responsibility, and a strong desire to learn. I am fully prepared to dedicate the required time and effort to deliver high-quality results.

During the summer, I will be completely available and focused on GSoC, without any conflicting internships or major commitments. This allows me to dedicate my full attention to development, testing, discussions, and continuous improvement.

Beyond GSoC, I aim to continue contributing to EROFS by improving tools, fixing issues, and supporting future contributors. I see this as an opportunity not just to complete a project, but to grow as a developer and contribute to meaningful open-source systems.

WHY ME?

I believe my technical background, adaptability, and problem-solving mindset make me a strong fit for this project. My experience across systems, Rust, testing frameworks, and CI/CD workflows has helped me understand how to build reliable and maintainable solutions.

I am comfortable working on system-level problems and enjoy understanding how things work internally. I also learn quickly and adapt based on feedback, which helps me improve continuously during development.

COMMENTS

• **Dedication to Quality:**
"I am excited to build tools that improve reliability and make EROFS more robust in real-world scenarios."

• **Team-Driven Approach:**
"I look forward to learning from experienced maintainers and contributing meaningful improvements."

• **Impact-Oriented Mindset:**
"I am motivated by the real-world impact of EROFS and the opportunity to strengthen a widely used Linux filesystem."

FINAL THOUGHTS

I am genuinely excited about the opportunity to contribute to EROFS and collaborate with its maintainers. The idea of building tools that improve filesystem reliability and help detect critical issues is both challenging and meaningful to me.

Through this project, I aim to deliver a robust fuzzing and testing framework that enhances the stability and security of EROFS.

I am committed to delivering high-quality code, clear documentation, and consistent contributions throughout the GSoC period.

For me, GSoC is not just a program—it is an opportunity to become part of a community building impactful systems used at scale.

• I am eager to collaborate, learn from experienced developers, and improve my approach.
• Beyond GSoC, I would like to continue contributing to EROFS and similar system-level projects.
• This project aligns perfectly with my interest in systems programming and building reliable software—and I am ready to give it my full effort.
