International Business Academy

In collaboration with

Bsc (Hons) in Informatics

Open Source Development

Student Name:

Asim Karki

Word Count:1546

16/03/2025

Module Leader:

Niels Müller Larsen

KOL389COM

Table of Contents

Introduction 2
What is Open-Source Development? 3
Tools of Open Source: Git and GitHub 4
The Role of Automation: GitHub Actions 4
Collaborative Development with Issues and Pull Requests 5
Why Open Source Matters for Me 5
1. Skill Development 5
2. Portfolio Building 5
3. Learning from Real Codebases 6
4. Community and Culture 6
Challenges Faced and Lessons Learned 6
The Future: My Career and Open Source 6
Conclusion 7

Introduction

Open-source development has fundamentally reshaped the landscape of software engineering, redefining how software is created, shared, and improved. At its core, open source embodies a philosophy of transparency, collaboration, and community-driven progress. It invites developers from around the world to contribute to projects, exchange ideas, and collectively solve complex problems. This global approach not only accelerates innovation but also ensures that software solutions remain adaptable, resilient, and accessible to all.

As a final-year Informatics student, engaging in open-source development represents more than just a technical undertaking, it is an immersive experience into the real-world practices and guiding philosophies that power today's software ecosystems. It provides an opportunity to step beyond the classroom, connect with active developer communities, and understand the dynamics of distributed teamwork, version control, peer review, and continuous integration.

In this essay, I will explore the core principles that define open-source development, highlighting its emphasis on openness, meritocracy, and shared ownership. I will also discuss its practical application within the context of my coursework, reflecting on the challenges and learning experiences encountered while contributing to an open-source project. Finally, I will examine the long-term relevance of open source to my future career in software engineering, and how this paradigm continues to influence not just how we write code, but how we collaborate, learn, and innovate in a rapidly evolving digital world.

What is Open-Source Development?

Open-source software (OSS) refers to software whose source code is available publicly for anyone to view, use, modify, and distribute. Unlike proprietary software, where code is locked behind licenses and corporate control, OSS encourages transparency and community involvement.

The foundational philosophy behind open source is built upon four freedoms outlined by the Free Software Foundation:

1. The freedom to run the program for any purpose.
2. The freedom to study how the program works and modify it.
3. The freedom to redistribute copies.
4. The freedom to distribute modified versions.

Popular open-source licenses, such as the MIT License, GNU General Public License (GPL), and Apache License, ensure legal clarity while enabling this level of openness. Open source has played a pivotal role in the success of modern technologies — from the Linux operating system and the Python programming language to web servers like Apache and frameworks like React.

Tools of Open Source: Git and GitHub

Version control is the backbone of collaborative development, and Git has become the de facto standard in this space. Developed by Linus Torvalds in 2005, Git allows developers to track changes, collaborate without conflict, and manage complex codebases.

GitHub, built on Git, is more than a code-hosting platform. It serves as a collaborative environment where teams across the globe can:

- Review each other’s code through Pull Requests (PRs)
- Track tasks and bugs via Issues
- Automate testing and deployment via GitHub Actions
- Maintain documentation using Wikis and markdown files

In this portfolio project, Git and GitHub were used extensively to manage and track development of the Python-based Yatzy game. Branching was used to isolate features and fixes, and commits were made to document changes logically. Pull Requests facilitated code review and quality assurance.

The Role of Automation: GitHub Actions

Automation is a key principle in modern software development. Continuous Integration/Continuous Deployment (CI/CD) pipelines help catch errors early and reduce manual intervention.

Using GitHub Actions, this project included a workflow to:

- Run tests automatically upon code push or pull request.
- Ensure that the new code does not break existing features.
- Maintain code quality and reliability.

This practice, although implemented on a small scale, mirrors real-world open-source practices. In large projects like Node.js or Django, every pull request is tested via automated CI pipelines before being merged. Learning to configure these tools prepares me for future work in professional software environments.

Collaborative Development with Issues and Pull Requests

One of the most powerful aspects of open source is collaborative problem-solving. Contributors can raise issues to highlight bugs or suggest enhancements. These are publicly tracked and linked to discussions, pull requests, and code changes.

In this coursework, I invited a peer to review my Yatzy class and raise an issue. Their observation led to the discovery of a logical error in the FullHouse () method. I responded by creating a new branch, fixing the issue, and merging it via a pull request. This mirrors the open-source model of community feedback and continuous improvement.

Moreover, documenting this process gave me insight into the importance of clear commit messages, detailed issue descriptions, and well-structured PRs all of which contribute to maintainable software.

Why Open Source Matters for Me

1. Skill Development

Working in open source environments sharpens key technical skills: Git workflows, automated testing, writing maintainable code, and understanding licensing. It also improves soft skills such as communication, code review, and problem solving.

Through this coursework, I have gained experience that is directly transferable to industry settings. Employers increasingly look for candidates who are comfortable working in version-controlled, collaborative environments. By using GitHub and following structured development practices, I have positioned myself for smoother entry into professional roles.

2. Portfolio Building

Contributions to open source projects serve as living proof of one's skills. They go beyond CV lines they are visible, measurable, and evaluable. This coursework itself is structured to simulate an open-source project, giving me a strong portfolio artifact.

In the future, I plan to contribute to projects that align with my interests such as AI, data visualization, or education tech. My GitHub presence will act as a professional profile, showcasing both code quality and collaborative etiquette.

3. Learning from Real Codebases

Reading source code from mature open-source projects exposes me to architectural patterns, best practices, and problem-solving strategies used by experienced developers. Platforms like GitHub and GitLab offer a window into real-world development, which is often more complex and nuanced than textbook exercises.

This exposure helps bridge the gap between academic knowledge and real-world application.

4. Community and Culture

The open source community is one of the most vibrant learning environments. From Stack Overflow answers to detailed GitHub discussions, developers around the world share knowledge freely. Participating in these communities fosters a culture of mutual respect, shared growth, and lifelong learning.

In the process of this project, I have interacted with peers, reviewed each other's code, and engaged in constructive technical dialogue all of which reflect the open source spirit.

Challenges Faced and Lessons Learned

While the open source approach offers many benefits, it also poses challenges:

- **Complexity**: Understanding unfamiliar codebases or debugging issues can be overwhelming.
- **Communication Gaps**: Writing clear issues and PRs takes practice and clarity.
- **Licensing Confusion**: Navigating different open source licenses can be tricky.

Through this project, I learned to address these challenges with patience and documentation. I also realized the importance of breaking tasks into manageable pieces and testing frequently.

The Future: My Career and Open Source

As I look ahead to my professional journey, I plan to pursue a career in software engineering with a particular focus on backend systems or data engineering. These areas are not only foundational to modern software infrastructure but are also heavily influenced and supported by open source technologies. Tools such as PostgreSQL, Redis, Apache

Kafka, and TensorFlow are just a few examples of the powerful, community-driven platforms that have become industry standards. These tools are not just utilities they represent the culmination of years of collaborative effort, innovation, and shared expertise within the open source community.

Knowing how to interact with these tools, read their documentation, contribute fixes or plugins, and possibly join their contributor communities will be a tremendous asset. Moreover, I hope to eventually mentor others in the same way I was helped keeping the spirit of open source alive.

Furthermore, my long-term goal is not just to benefit from open source, but to give back to it. I aspire to eventually take on a mentorship role, guiding newer contributors in the same way that I was supported during my early experiences. Open source thrives on the spirit of mutual support and knowledge sharing, and I am committed to upholding that ethos throughout my career. Whether through code contributions, documentation improvements, or helping others get started, I hope to be an active participant in sustaining the open source movement and ensuring its continued relevance and vitality in the software engineering world.

Conclusion

This coursework offered a practical introduction to open source development from using GitHub and writing modular code to automated testing and peer collaboration. More than a project, it was an immersion into a culture of openness, transparency, and community learning.

This coursework provided a valuable and hands-on introduction to the world of open source development, bridging the gap between theoretical understanding and real-world application. Through the various components of the project, I was able to engage directly with industry-standard tools and practices from managing repositories on GitHub and writing clean, modular code to implementing automated testing frameworks and collaborating effectively with peers. Each of these elements played a crucial role in reinforcing the technical and interpersonal skills that are essential in modern software development.

In a world where software is increasingly collaborative, the ability to contribute to and thrive in open source ecosystems is a critical skill. Open source is not just a development model it is a mindset, and one I plan to carry with me into my career and beyond.