BINGE - Browser extension for detailed information on Streaming Platforms

Eshita Arza
North Carolina State University
Raleigh, North Carolina, USA
sarza@ncsu.edu

Rahul Kalita
North Carolina State University
Raleigh, North Carolina, USA
rkalita@ncsu.edu

Vignesh Muthukumar
North Carolina State University
Raleigh, North Carolina, USA
vmuthuk3@ncsu.edu

Isha Gupta
North Carolina State University
Raleigh, North Carolina, USA
igupta@ncsu.edu

Luis Gabriel Delassantos
North Carolina State University
Raleigh, North Carolina, USA
lgdeloss@ncsu.edu

Kiran Teja Tummuri
North Carolina State University
Raleigh, North Carolina, USA
ktummur@ncsu.edu

ABSTRACT

Binge is an chrome extension that aims to provide quality and essential information about video content on streaming platforms to users which will help them get the best streaming experience. According to [1] 63% of users look out for details like ratings, people's reviews, critic's reviews and cast information on popular platforms available the internet like IMDB, Rotten Tomatoes, etc., before deciding to watch a movie or a series. This extension saves a lot of time and effort for the users and provides a one stop place to get quick access to required information about the content. The first section allows readers to get familiarized with the application and how to contribute to the code base [2]. This paper clearly delineates the 5 Linux Kernel Best Practices used to develop state of the art software engineering applications and the connection between each practice and the implementation in our application.

CCS CONCEPTS

• **Software Development Practices - Linux Kernel**; • **Binge** → Streaming information extension;

KEYWORDS

software engineering, best practices, browser extension, online streaming

1 INTRODUCTION

The Linux kernel which has existed over decades serves as the best example for the world of software developers to focus on the way of software development. Crossing the era of technological innovations and invasions, Linux still maintains its standard of performance and stands as a pioneer because of the extensive development practices that the institution believes and implements. By far, the Linux kernel best practices has been accepted world wide and it serves as a benchmark for developing and evaluating quality software products.

The 5 principles that Linux describes as the best practices are :

- Zero Internal Boundaries
- No Regression Rule
- Consensus Oriented Model
- Distributed Development Model
- Short Release Cycles

This application is primarily developed with a notion to adhere with the Linux best practices. The paper elucidates each of the individual metric that is practiced and provides a solid proof of implementing these practices in various development stages of the project. As novice developers into the field of software engineering, these practices lay a strong foundation to always look out for the best practices elsewhere while developing applications in the industry.

2 LINUX KERNEL DEVELOPMENT BEST PRACTICES

2.1 Zero Internal Boundaries

Zero internal boundaries means that every team member involved in the project gets to access and use common resources, tools, information and accessibility privileges throughout the project. This means that there should be no barrier for any developer towards the contribution to the project. This was ensured at the very beginning of the project during the idea phases that everyone works on what commonly aligns with the prospect of the team.

To achieve "Zero Internal Boundaries", we made a GitHub organization and gave equal access to everyone. This helped us negate the single person "Admin" access on user-based repositories. We have created our project "Binge" on the organization GitHub account.

Our web extension was built using vanilla JavaScript which had one of the best open source support available and every member had accessibility to the resources. Every member also has access to all components used in or related to the project, such as APIs, Issues, Roadmap, Release and versioning.

2.2 Consensus Oriented Model

Consensus oriented model suggests that any decision that is taken amongst the developers about the project must be a collective one. This ensures integrity among the developers and there will be a streamlined thought about the goal amongst the entire team throughout the project. The Linux kernel best practices method emphasizes this because every member brings diverse perspectives to the table and the outcome of each decision must be analysed clearly and effectively to deliver the goals in a desirable model.

We started our project with a collective consensus. All the members brought some ideas to the table and we all agreed on making Binge as our final project. Our project workflow strictly followed this principle by having a group moderator and every person was given

a chance to put forward their views in sequential order. We agreed upon each and every task and requirement before ending the discussions. Also before handling issues and bugs in other piece of code, the complete consent from the original developer was sought to ensure the concrete structure of the existing code does not get affected in any way.

2.3 No Regression Rule

The main focus of the 'No Regression Rule' is that, with every new code or feature addition, the existing functionality of the application should not get affected. Since feature addition and code refactoring is a constant process that happens in any development environment, utmost care has to be taken to not eliminate any required functionality. Also, it is expected from the developers to address these issues quickly and bring the application back and running. We consider this as the most important principle in our development process as every individual who works on independent features should not hamper the other existing features in the application. This was ensured by having strong test cases and a code coverage mechanism which checks for minimum threshold values to be passed for each pull request and code commit that was made.

2.4 Distributed Development Model

Distributed development model enforces the idea of distributing the entire development workflow amongst all the members of the team for a seamless development process. The Linux best practices [3] enforces this to ensure that every team member has an opportunity to equally contribute on the project. Also, the workload is equally divided amongst everyone and this also provides a way to bring in novel ideas and innovations to make the development process more interesting. During the initial phase of this project, there were multiple rounds of discussions that happened as a team. This allowed each other to understand the strengths and weaknesses of the team and divide the work accordingly. Since BINGE is a browser oriented extension, this involved work on both front end and back-end framework. Eventually the tasks were equally split among all the individuals to work on various features and sub tasks were cross assigned for performing validation and integration testing. Also we practiced the process of code reviews on each other's work and ensured to follow coding standards throughout the project like nomenclature of the variables, designing the directories, writing doc-style code comments for every function etc. The retrospective insight about having had a distributed development model in our development workflow ensured that there was no spillover or dependency issues on tasks as each module was handled independently. To coordinate all of this seamlessly, a strong communication and coordination was expected amongst individuals. We had a Whatsapp group created and followed a scrum kind of practice where every individual drops a quick update on their work progress and we had weekly sync ups to cover larger parts of development.

2.5 Short Release Cycles

Short release cycles are usually practiced to roll-out features in an iterative and periodic manner to the market. It is believed that if a software reaches the world even with a minimum viable product, then the scope of the project tends to increase drastically. Also, short release cycles provide to the developers and stake holders about the changes and extensions that can be added to the features. Short release cycles also allow organizations to build stable versions of software and can address issues quickly. Continuously integrating new code enables fundamental changes to the code base to be introduced without causing large interruptions. In this project we have committed to continuously release code phase-wise, over the entire duration available for the project development. Over the period of one month, we had constant pull reviews merged to the code base and commits were made per feature and bug which ensured that the entire workflow of the code did not get hampered.

3 REFERENCES

(1) https://www.statista.com/statistics/898999/reading-reviews-before-viewing-movies-united-states/
(2) CSC 510 - Group 7 - Fall 2021 : https://github.com/NCSU-Group7-SE2021/Binge
(3) Timothy Menzies. 2021. https://github.com/txt/se21/blob/master/docs/proj1rubric.md