SE Project 1 : PackTravel (Group : 8)

Ameya Chavan
North Carolina State University
aachava2@ncsu.edu

Amisha Bipin Waghela
North Carolina State University
awaghel@ncsu.edu

Aoishi Das
North Carolina State University
adas23@ncsu.edu

Kunal Shah
North Carolina State University
kshah24@ncsu.edu

Swarnamalya Mohan
North Carolina State University
smohan7@ncsu.edu

ABSTRACT

Most of the international students do not have a car to travel off-campus and rely mostly on the Wolfline. But what if someone wants to travel outside Wolfline's limit? Well they can collaborate on PackTravel to travel off-campus by a cab, rental car, etc.Here we are going to draw a connection between Linux Kernel Best Practices and points mentioned in the rubric to evaluate if our project follows the best practices or not.

KEYWORDS

Linux Kernel Best Practices, Software Engineering, PackTravel

1 ZERO INTERNAL BOUNDARIES

Having zero internal boundaries is one of the Kernel Development Best Practices.Zero internal boundaries mean that although different members are working on different parts of the project they can still contribute and make changes in other parts. In our project the following points show that we have implemented Zero Internal Boundaries :

- Workload is spread over the whole team (one team member is often Xtimes more productive than the others... but nevertheless, here is a track record that everyone is contributing a lot) : Everyone has committed and contributed equally.
- Number of commits : There's a lot of commits made frequently by all the members
- Number of commits: by different people : Different members have made commits in different areas
- Evidence that the whole team is using the same tools: everyone can get to all tools and files : Everyone has used the same tools and all the files are accessible by all the members
- Evidence that the whole team is using the same tools (e.g. config files in the repo, updated by lots of different people) : The whole team has been using the same tool
- Evidence that the members of the team are working across multiple places in the code base : Members can be seen making commits in different areas of the code base
- Is your source code publicly available to download, either as a downloadable bundle or via access to a source code repository? : Yes it is publicly available for download. It can be either cloned or downloaded as zip.
- Are e-mails to your support e-mail address received by more than one person? : Yes. This ensures that every member have equal access.

All the members have used the same tools for coding. We have made use of virtual environments and the requirements.txt file was installed which ensured that everyone has access to the same libraries and models.

2 SHORT RELEASE CYCLES

Short Release Cycles makes code available quickly in a stable release and integrating new code on a constant basis also helps to bring in even fundamental changes with minimal disruption. In the early days larger release cycles used to be frustrating because it delayed getting features out to the users. Besides it also meant integrating huge amount of codes at once.

- Short release cycles : Regular commits and releases have been made
- Number of commits : There were a lot of commits made quite regularly by the team members
- Number of commits: by different people : All members made commits and contributed to different areas.

3 CONSENSUS ORIENTED MODEL

Consensus Oriented Model is one of the Kernel Development Best Practices. It ensures that everyone have a say in the development and changes are made only when there is a consensus .This is important for open source development as this will ensure that everyone has a say in the development process and no one's idea is ignored. In our project the following points show that we have tried to implement this :

- Evidence that the whole team is using the same tools: everyone can get to all tools and files : Everyone is using the same resource and they have access to all the files and models
- Are e-mails to your support e-mail address received by more than one person? : Everyone has the credentials to the email and so anyone can attend to the issues or queries
- Issues are discussed before they are closed : Issues are discussed over chat channel and physical meetings
- Chat channel: exists : Slack channel exists to discuss ideas and decide on how to proceed

4 THE NO REGRESSION RULE

The No Regression Rule means that if a given kernel works in a specific setting, all subsequent kernels must work there, too. This will ensure the users that any updates to the software won't break their system. It is one of the Kernel Development Best Practices.In our project the following points show that we have tried to follow this :

- Use of version control tools : GitHub was used ensuring that people need to make branches and push their changes

- Test cases exist : Test cases will ensure that the new code need to pass the tests
- Test cases are routinely executed : This ensures that the changes added are passing the test cases
- The files CONTRIBUTING.md lists coding standards and lots of tips on how to extend the system without screwing things up : This serves as a guideline for anyone interested in making a contribution without creating any problems in the existing code base

5 DISTRIBUTED DEVELOPMENT MODEL

Distributed development Model is a one of the best software development model in which a team spread across different locations can collaborate on applications or various software. The work is split into different functionalities and different members work on different areas in accordance to their strength or familiarity finally integrating the parts together. In our project the following points show that we have tried to follow this :

- Workload is spread over the whole team : The workload is spread across the whole team with members making contributions to different areas
- Chat channel: exists : Slack was used as a medium of sharing information, ideas and resources. The team even met up physically to discuss ideas and progress.
- Evidence that the whole team is using the same tools: everyone can get to all tools and files : Everyone used the same tools
- Evidence that the members of the team are working across multiple places in the code base : Every member has contributed to multiple areas
- Do you accept contributions (e.g. bug fixes, enhancements, documentation updates, tutorials) from people who are not part of your project? : Yes
- Do you have a contributions policy? : Yes and the CONTRIBUTING.md file states all the necessary steps
- Is your contributions' policy publicly available? : Yes