Research Buddy

Kriti Khullar
kkhulla@ncsu.edu
North Carolina State University
Raleigh, NC, USA

Muskan Gupta
mgupta23@ncsu.edu
North Carolina State University
Raleigh, NC, USA

Suneha Bose
sbose2@ncsu.edu
North Carolina State University
Raleigh, NC, USA

Urvashi Kar
ukar@ncsu.edu
North Carolina State University
Raleigh, NC, USA

Usman Mahmood Khan
ukhan3@ncsu.edu
North Carolina State University
Raleigh, NC, USA

ABSTRACT

Research Buddy is a chrome extension which helps the user reduce their cognitive load while researching on the browser by providing them with resources that they would spend extra time and effort looking up by themselves. Suppose the user is reading a research paper, and they read a terminology which is unknown to them. In usual circumstances the user would open a new tab to google the term and then go through sources such as Wikipedia, Youtube and others to understand the new term. Research Buddy lets the user highlight the word which then shows a pop-up that gives them the Wikipedia definition with a link to the page, and links pointing to the top 5 related papers as well as the search results in Youtube, Github, Udemy, Google Scholar and Coursera.

ACM Reference Format:
Kriti Khullar, Muskan Gupta, Suneha Bose, Urvashi Kar, and Usman Mahmood Khan. 2018. Research Buddy. In *Woodstock '18: ACM Symposium on Neural Gaze Detection, June 03–05, 2018, Woodstock, NY*. ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/1122445.1122456

1 KERNEL DEVELOPMENT BEST PRACTICES

Linux Kernel Development practices are a great model for developers to implement to their own development journey. With it being Open Source its regularly updated and modernized with the passing time

2 SHORT RELEASE CYCLES

Research Buddy is a short project and yet in the commit history, it can be seen that there is a slow but constant progress in the growth of the code base. And with each commit, the update is already in the main branch making it available for use to anyone who wishes to use it. Short Release Cycles are intended to make our software better updated to the current requirements of the customer base. Earlier, the updates to a particular code repository was done at an interval of 4-6 months. This created multiple problems such as a huge part of code had to be integrated all at once, which meant a lot of testing to be done right before the delivery, it also held off updates to the software for a long time leaving the users waiting for the newer updates. With Short release cycles, the code updates are done on a regular basis, this time frame could be either immediate or range up to 2 weeks. Agile methodology uses Sprints as a way to keep the release cycles every two weeks. This way the deliverables are in production environment every two weeks.

3 DISTRIBUTED DEVELOPMENT MODEL

There are always multiple aspects of a software piece. The frontend, design, backend, database, implementation, testing, documentation, presentation and so many more. When a team works on a software, these aspects are best achieved by using the distributed development model. Dividing the tasks among the team makes sure that each task is assigned to at least one member and no task is left out. This also helps in keeping track of all the tasks being done. The task division can be done in two ways, one way is based on the skill set of each teammate, so everyone works on the part they are best at, and the other is we divide in such a manner that each member contributes equally to every part. Our distributed model for Research buddy has been a mix of the two, we started off focusing on parts that we are better at and then as we started getting involved in all parts eventually. Number of commits is one indicator of how much each team member has contributed, but it can't always be the right metric to check the contribution of a developer in the project. There's many non functional aspects to a software which are not tracked via tools.

4 CONSENSUS ORIENTED MODEL

It is essential that before a particular code change is merged into the code base, the change is discussed between all the team members or at least the concerned members who would be affected by it. Consensus Oriented Model aims to implement this in teams. Pull requests are a feature which help implementing this model as well, the reviewers of the code base get to hold off the merge until they are satisfied with the change. They let the developer know about their opinions and requirements via comments. In Research Buddy, there were discussions on chat channels as well as in-person, before each change was implemented. It can feel like this takes up extra time before a delivery but it helped in the collaborative satisfaction of the team with respect to the code repository.

Woodstock '18, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

5 THE NO-REGRESSIONS RULE

Softwares are updated throughout their lifecycle. The No regression rule takes care that at each version of the software, there is no regression, which means the code doesn't break. This keeps the software reliable and makes the software more usable and adds to the longevity of its life in the market. And even when there is a regression, it becomes the top priority for the team to fix it. Resource buddy is still not in market, but the team has striven to keep it from breaking and work seamlessly across its scope.

6 ZERO INTERNAL BOUNDARIES

Use of version control tools : We have used Github as our version control tool. Also used a same platform vs code to implement the basic functionalities.

7 CONCLUSION

Research Buddy, is in its very initial stages, and already is of great help to users. It enhances the experience of browsing on the web in a simple and efficient manner. The development of this extension was done in a short span of time and implementation of the Linux Best Practices made the development process more organised and kept chaos in the bay.