Track-a-day

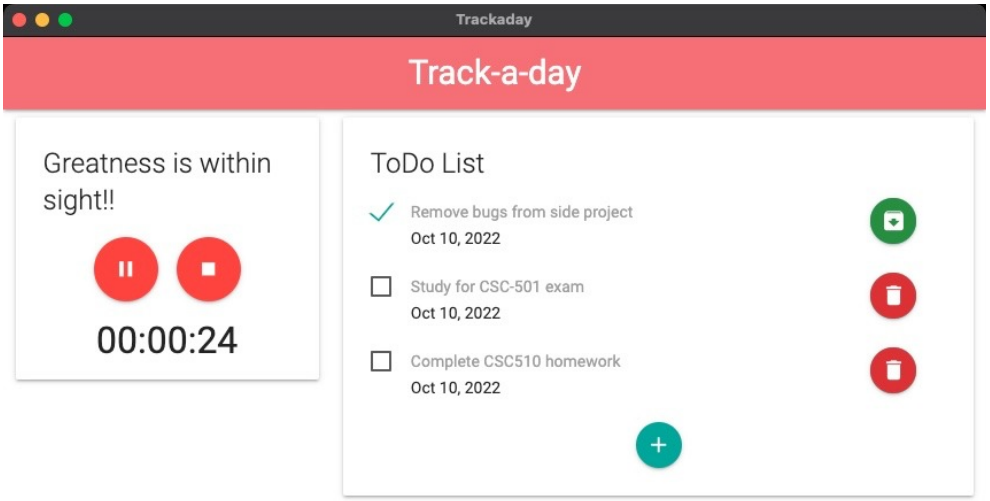
Arnab Datta
NC State University
Raleigh, United States
adatta@ncsu.edu

Atharva Gole
NC State University
Raleigh, United States
amgole@ncsu.edu

Dakshil Kanakia
NC State University
Raleigh, United States
drkanaki@ncsu.edu

Dhanya Dasari
NC State University
Raleigh, United States
ddsari@ncsu.edu

Vishal Sharma
NC State University
Raleigh, United States
vsharma7@ncsu.edu



✓ Remove bugs from side project
Oct 10, 2022

□ Study for CSC-501 exam
Oct 10, 2022

□ Complete CSC510 homework
Oct 10, 2022

Figure 1: Home Page of ou application.

ABSTRACT

The following is a report based on how the Linux best practices are followed for Group 38 of the first project. Our project is titled Track-a-Day, a desktop application that can be utilized to keep track of daily tasks and get analysis reports.

CCS CONCEPTS

• **Computer systems organization** → **Embedded systems**; *Redundancy*; Robotics; • **Networks** → Network reliability.

Unpublished working draft. Not for distribution.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
Conference acronym 'XX, June 03–05, 2018, Woodstock, NY
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

KEYWORDS

software engineering, Tracking, best-practices, analysis

ACM Reference Format:
Arnab Datta, Atharva Gole, Dakshil Kanakia, Dhanya Dasari, and Vishal Sharma. 2018. Track-a-day. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (Conference acronym 'XX)*. ACM, New York, NY, USA, 2 pages. https://doi.org/XXXXXXX.XXXXXXX

1 SHORT RELEASE CYCLES

The first Linux Best Practice is short release cycles. New features can be added. Some minor bugs can be fixed. This is a great process for consumers since the product keeps upgrading. This also helps developers as they have the flexibility to test their code and solve minor bugs in the next version. This can also be used to test new features and if they turn out good, they can be released in following versions.

Our group follows this model and Track-a-Day has one alpha release. This release focuses on the structure of code along with the functionality.

2 DISTRIBUTED DEVELOPMENT MODEL

n distributed development model, theres no dependency on just a single person or writing code and approving what gets addded into the codebase. Rather the tasks are split evenly among all the group members. An advantage is that all the group members have a working idea of what is in the codebase. The addition of new features is not just dependent on a single person managing everthing.

Our group handles this like – 1) the development of the project was divided as a front end team and a backend team to increase the pace of the project, 2) when someone pushes a request it cannot be merged until it is approved by another member, 3) it is best if for a specific technology a specific person reviews and accepts the pull request eg- if someone is working on a AngularJS, they include Arnab since he has the most experience with it similarly if someone is working on ElectronJs they include Vishal.

Evidence for this can be found in project issue and pull requests.

3 CONSENSUS ORIENTED MODEL

The consensus-oriented model tells us that no group of users should block another group of users from contributing. It is a model where people working together to reach maximum agreement among themselves. The changes and modifications must be validated with developers before completely implementing them into the system.

To achieve this model our team first decided the goal. We brainstormed ideas for the project depending on the knowledge and experience each team member has. Multiple project ideas were proposed then we collectively decided to move forward with this project. We held discussions at the library and on group chat. The name, theme and UI was decided on a virtual meet.

At the time of development reviewing before merging helped us achieve this model of the project. If the reviewer indicated an error or suggests a change, the required modifications were made.

The evidence of this is visible in the pull requests and group chat.

4 THE NO-REGRESSION RULE

The No-regression rule is about moving forward and side by side maintaining the older versions of the project. If there are upgrades in the system, the users should still have all the functions of the older versions.

We follow this rule by not implementing any piece of feature that are not finished. For instance we wanted to add a certain feature, but due to time constraints we were unable to completely integrate it with the existing project.

If that half complete feature were to be added in the project, the users would have started using that feature and would face some bugs and this would resent the user.

The evidence of this rule comes from the code, design and pull requests.

5 ZERO INTERNAL BOUNDARIES

Zero internal boundaries suggests that everyone has the freedom to maks changes in anypart of the project. This way new features can be added by any developer, an issue can be resolved by any developer. This is similar to a distributed model. To achieve this, all the team members should have access to all the development tools.

Our team implemented this in – 1) the repository was public with which all members had the ability to access and make changes in the codebase.

Evidence for this rule can be found in instructions for the project README and some particular packages can be found in requirements.txt

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009

Unpublished working draft. Not for distribution.