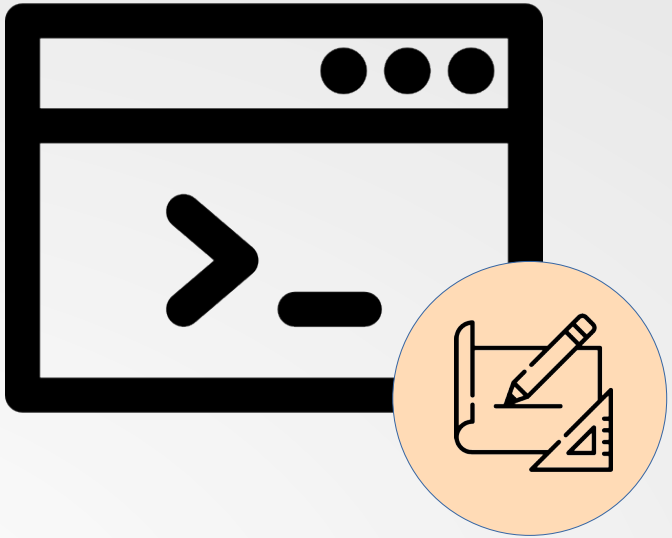
GNOME.ASIA Summit 2020

UX Design of CLI Application

Adityo Pratomo
@kotakmakan



GNOME.ASIA Summit 2020

UX Design of CLI Application

github.com/lunchboxav/ux-cli

Adityo Pratomo

UX designer, focusing on developer-centric product.

Also does cycling, skateboarding and playing Street Fighter.

About Me

Product Design at Tetrate

We build service mesh solution for enterprise, based on open source projects such as Istio, Envoy and Apache SkyWalking.

visit us at: tetrate.io

A quote from the past

> “It’s 2007, who wants to use CLI application?”
>
> - my professor

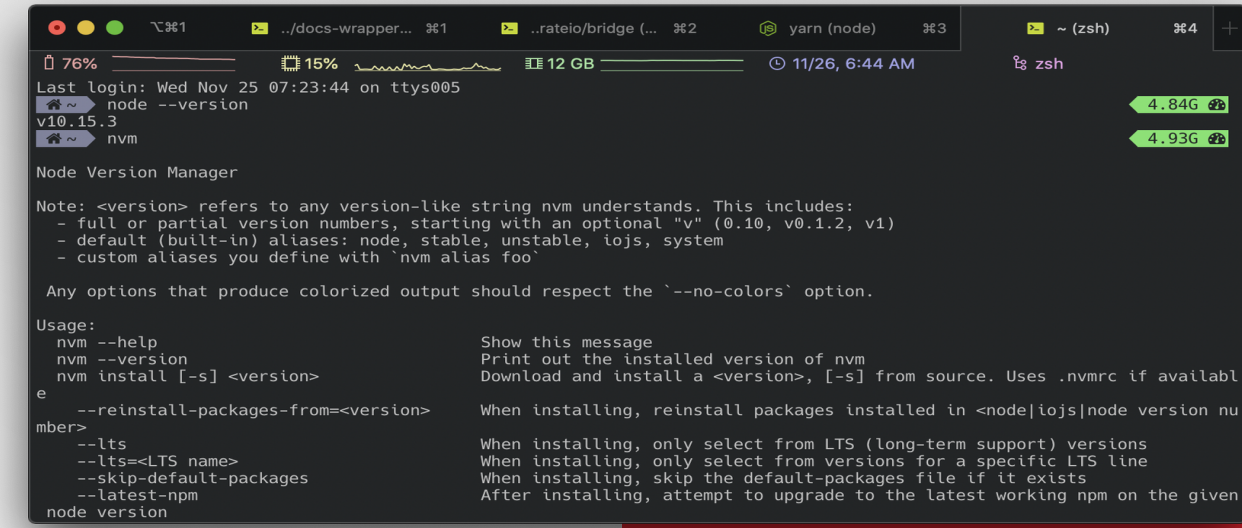
Fast forward to today

CLI Tools are everywhere

Front end developers use npm, yarn, nvm, webpack, etc. to build and manage their project

Staple CLI tools like ls, and grep are rebuilt in Rust to enhance the terminal experience

CLI Tools are having a renaissance



Why we love CLI Application?

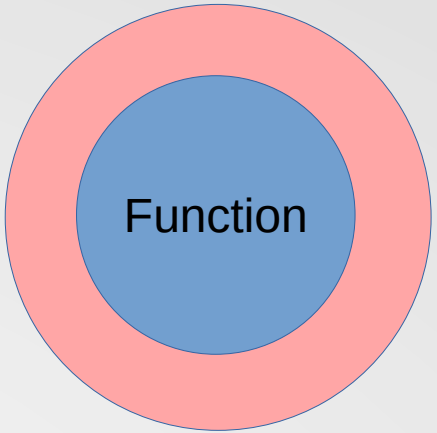
For specific application,
it just works!

Low resource consumption

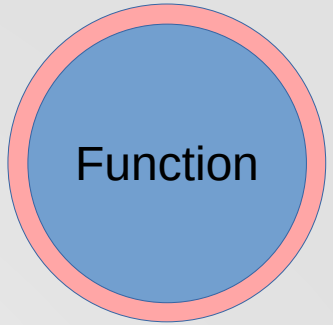
Makes you command the computer directly, with no visual abstraction

Opportunities for automation

Intuitive, text based interaction



GUI app with visual abstraction layering function



CLI app with thin text abstraction layering function

How about the UX?

Mental model alignment

Contextual match

Learnability

Memorability

Error tolerance

Clear feedback

How about the UX?

Mental model alignment	Yes
Contextual match	Yes
Learnability	Yes, but largely depends on help
Memorability	Depends on help
Error tolerance	Depends on the design
Clear feedback	Depends on the design

How can we improve?

Photo by Markus Winkler on Unsplash

Some problems

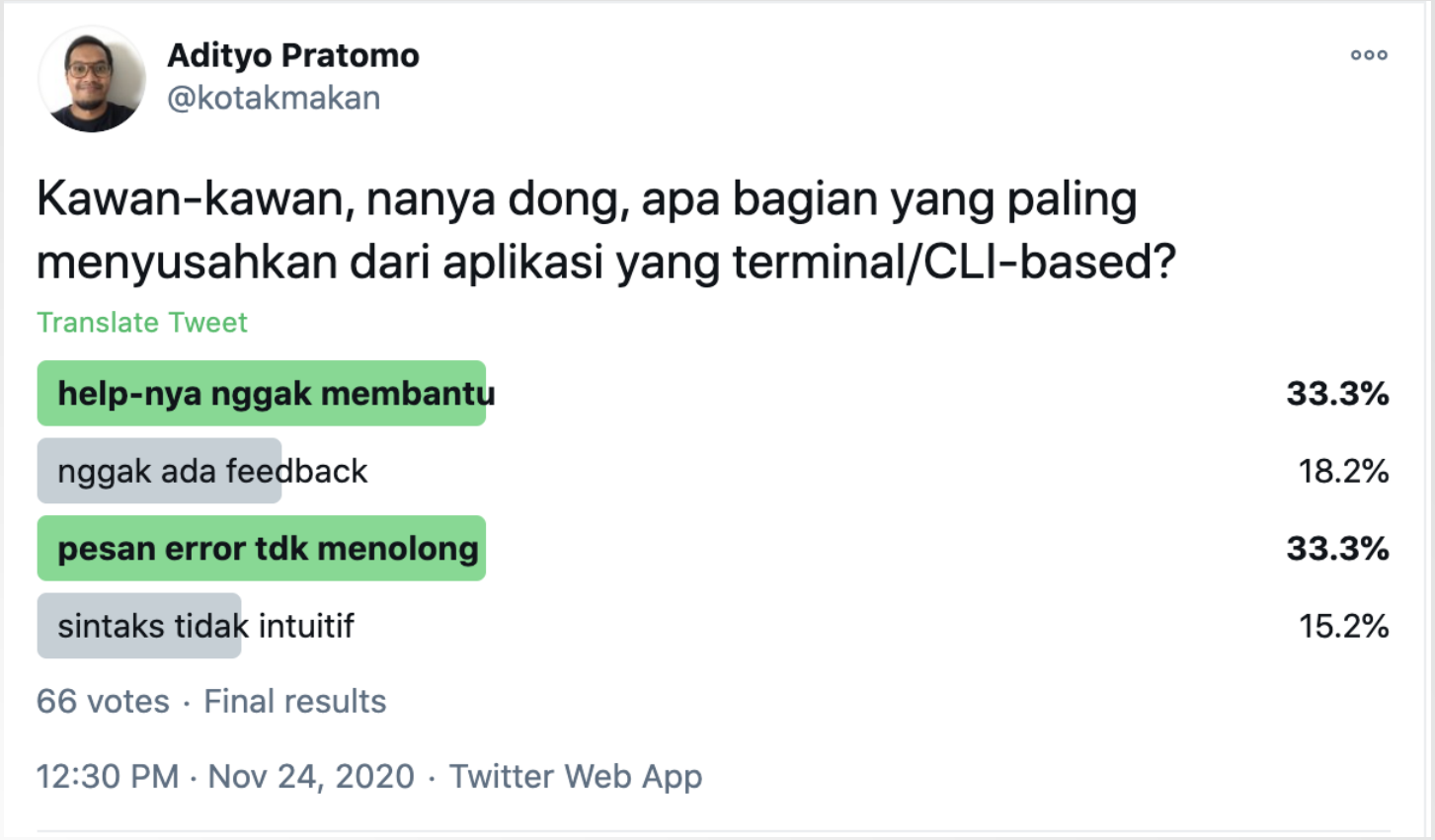
Unhelpful help

No feedback

Unhelpful error message

Unintuitive syntax

Some problems



Adityo Pratomo
@kotakmakan

Kawan-kawan, nanya dong, apa bagian yang paling menyusahkan dari aplikasi yang terminal/CLI-based?

Translate Tweet

help-nya nggak membantu	**33.3%**
nggak ada feedback	18.2%
pesan error tdk menolong	**33.3%**
sintaks tidak intuitif	15.2%

66 votes · Final results

12:30 PM · Nov 24, 2020 · Twitter Web App

Some problems

Unhelpful help	33.3%
No feedback	18.2%
Unhelpful error message	33.3%
Unintuitive syntax	15.2%

Design Challenge #1: help

> “How might we enable a CLI app user to explore and discover how to use that app in more efficient manner, so that the overall experience can be much smoother?”

Design Challenge #2: error message

> “How might we enable a CLI app user to learn from the error message so that he/she can recover quickly and be more productive”

Designing Help screen: what it should provide

GNOME.ASIA Summit 2020

01

Quick usage

Provides example of usage, including 1 most important command, followed up by other possibly used commands

02

List of commands and flags

Shows all available commands and flags that the user can skim, including what format to use. Using color might be helpful.

03

Link to external resource

To enables the user to read from browser, so that we can provide a deeper instruction and more detailed information

Designing Help screen: prototype

To add new todo item, use the command:

```
$ rafa add todo “pick up book from library”
```

This will create a new todo item named “pick up book from library”

Afterwards, you can do many more things

```
rafa check todo  : mark todo as done
rafa uncheck todo: mark todo as not done
rafa change todo : update todo information
rafa remove todo : remove a todo item
```

To learn more about what rafa can do for you, please visit https://rafa.io/docs

Designing Help screen: prototype


```
To add new todo item, use the command:

$ rafa add todo “pick up book from library”

This will create a new todo item named “pick up book from
library”


Afterwards, you can do many more things

rafa check todo  : mark todo as done
rafa uncheck todo: mark todo as not done
rafa change todo : update todo information
rafa remove todo : remove a todo item

To learn more about what rafa can do for you, please  visit
https://rafa.io/docs
```

Core commands instruction

Follow up commands instruction

Link to external resource

Designing Help screen: prototype


```
Usage:
     rafa [command] [flags]=”argument” todo “item”

Available commands:
     change   change detail of a todo item
     check    mark a todo item as done
     list     display all todo items
     remove   remove a todo item
     uncheck  mark a todo item as not done
     version  display currently running version of rafa

Available flags:
         --date   date detail of a todo item
     -h, --help   help information of a rafa command
         --name   name detail of a todo item
         --status status detail of a todo item, either done or not done

Note: Both todo item and flag argument has to be written between single
or double quote
```

Designing Help screen: prototype


```
Usage:
     rafa [command] [flags]="argument" todo "item"

Available commands:
     change   change detail of a todo item
     check    mark a todo item as done
     list     display all todo items
     remove   remove a todo item
     uncheck  mark a todo item as not done
     version  display currently running version of rafa

Available flags:
         --date   date detail of a todo item
     -h, --help   help information of a rafa command
         --name   name detail of a todo item
         --status status detail of a todo item, either done or not done

Note: Both todo item and flag argument has to be written between single or
double quote
```

Basic usage syntax

List of available commands and flags

Important note

Designing Help screen: prototype


```
Full example scenario:

To change the name of a todo item
    rafa change todo “grab breakfast” --name=”cook breakfast”

To mark a todo item as done
    rafa check todo “cook breakfast”

To change the deadline of a todo item
    rafa change todo “cook breakfast” --date=”28/11/2011”
    rafa [command] [flags]=”argument” todo “item”
```

Designing Help screen: prototype


```
Full example scenario:

To change the name of a todo item
    rafa change todo “grab breakfast” --name=”cook breakfast”

To mark a todo item as done
    rafa check todo “cook breakfast”

To change the deadline of a todo item
    rafa change todo “cook breakfast” --date=”28/11/2021”
    rafa [command] [flags]=”argument” todo “item”
```

Thorough example commands, based on scenario

Designing Error screen: what it should provide

GNOME.ASIA Summit 2020

01

Clear error cause

Provides a easy to understand message about what causes the error, maybe mistyped or missing argument, or perhaps an unavailable command

Error code might also be helpful.

02

Possible fix

How a user can fix the error, what's the possible solution? This might not even be applicable 100%, but at least it should point the user towards a better direction.

03

Link to external resource

In case the possible cause and fix to the error can be provided in much more detail in a web page, then a link to that page should be provided

Designing Error screen: prototype


```
$ rafa add todo

[error, code: 20]: You did not provide name for that todo item.

[possible fix]: Please provide a name, written between double
quote, for example:

rafa add todo “buy milk”

To see detailed information about this error, please visit
https://www.rafa.io/docs/error/20
```

Designing Error screen: prototype


```
$ rafa change todo “cook breakfast” --date=”28/November/2021”

[error, code: 29]: The date argument is not following the
dd/mm/yyyy format

[possible fix]: Change the argument following the required format,
dd/mm/yyyy, for example:

rafa change todo “buy milk” --date=”15/04/2020”

To see detailed information about this error, please visit
https://www.rafa.io/docs/error/29
```

What might help us in implementation

CLI Grammar	A table that lists down all the possible commands, flags and objects, what they do and how they relate
Error map	A table that lists down possible error, codes, cause and possible fixes
Config file	Better to store the app's list of commands in easy to read and edit config file
Colored text	Use color to help user in navigating the messages, but always be mindful of accessibility.

If you want to do it in Rust

CLI Grammar	Any table editor
Error map	Any table editor, https://github.com/rust-cli/human-panic
Config file	https://github.com/clap-rs/clap https://crates.io/crates/libcli
Colored text	https://github.com/LukasKalbertodt/bunt/, https://github.com/SergioBenitez/yansi

To conclude

CLI application is a very powerful tool for developer. However, it’s still has a lot of room to improve.

Enhancing the day to day activity, starting from easy onboarding, to helpful vital screen such as Help and Error, can be a low hanging fruit in help its user to enhance productivity.

Thank You