1. Very light & introduction to the module
2. New module added in scaler
3. All classes will last ~ 2 hours.
4. The class names are messed up.
5. Its an attempt to transfer experience
6. You should practice after every class.
   Also, try adding few more features.
7. If you get stuck --
   (i) use google/chatgpt effectively
   (ii) you can help each other
8. Make you a more matured engineer.
   You should explore and figure out the answers when you're stuck.

Two types of people companies don't hire.

1. People who cannot follow instructions.
2. People who only follows instructions.

Today's agenda

1. Front end v/s Backend
2. Curriculum
3. Version control system.

Any app^n , Front end Backend.

Front end

People whoire using your app^n,
will interact via frontend.

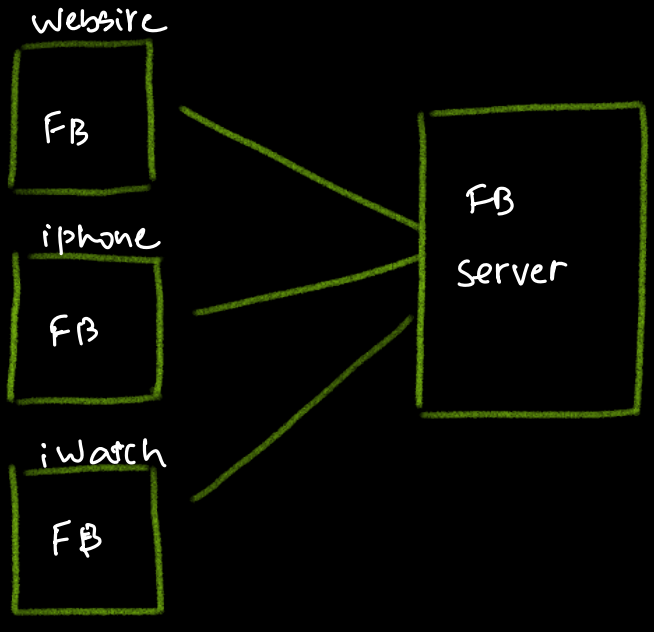
How can one interact with a Sim ?

* website / app^n / TV / Alexa / iWatch etc --

Anything that requests data from the server.

Backend (Server)

1. The component which stores / retrieves data required by taking a request from front end.
2. The server / backend owns the data.



Curriculum.

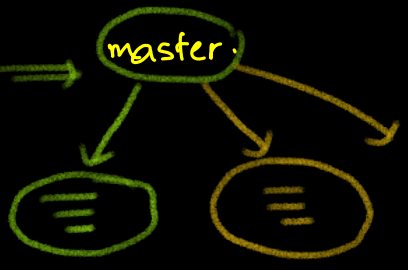
1. Intro
2. Git → 2-3 classes
3. Spring boot → 2
4. another server call
5. How to interact with db → 3-4 classes
6. unit testing → 3-4 classes
7. Authentication vs Authorization → 4 classes
8. AWS → 2-3 classes
9. Search, Sorting
10. Payment gateways
11. Spring cloud → 3 classes
12. Docker & Kubernetes.

ecommerce app^n.

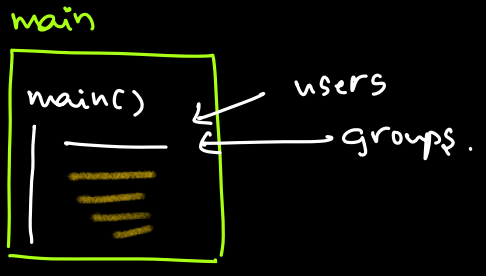
Hands-on.

Version control system (VCS)

1. You'll not have a single codebase where you keep everything
2. You'll not be the only one who contributes to the code.



Splitwise



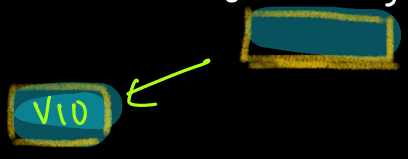
VCS : V_1 , V_2 , V_3.

Your codebase will have multiple versions, instead of we managing these versions we rely on VCS.

Why we need to keep track of versions

→ revert some change.

→ changes made by one person.



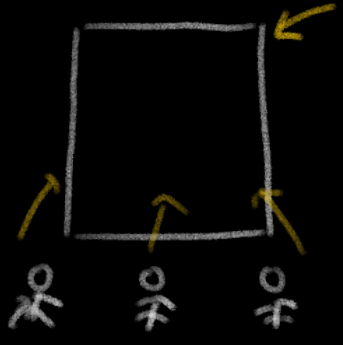
VCS will help you to understand how a codebase looked from starting point till today.

Types of VCS

1. Centralised VCS

A single server is maintaining the codebase.

ex: SVN



issue: Single point failure.

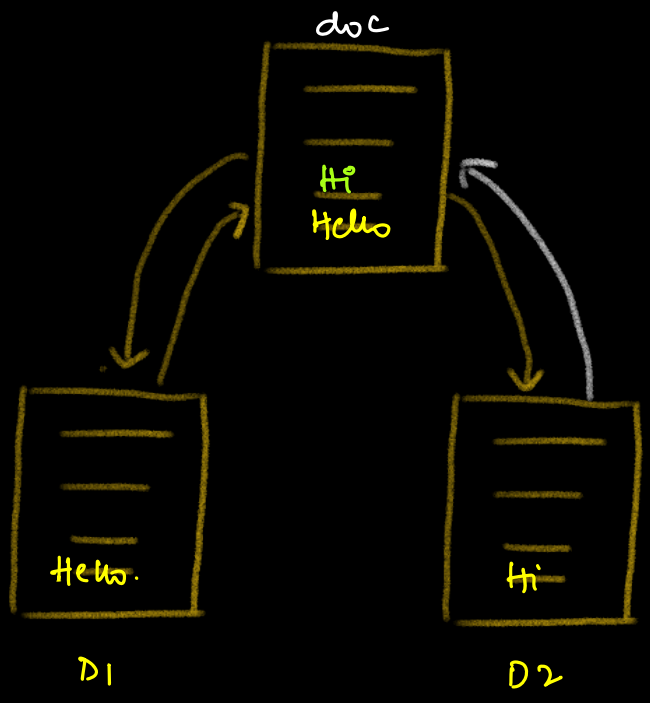
2. Distributed VCS.

Every team member should be able to work independently.

Every person will have a copy of their changes & their history.

Local history for every change will also be maintained.

ex:

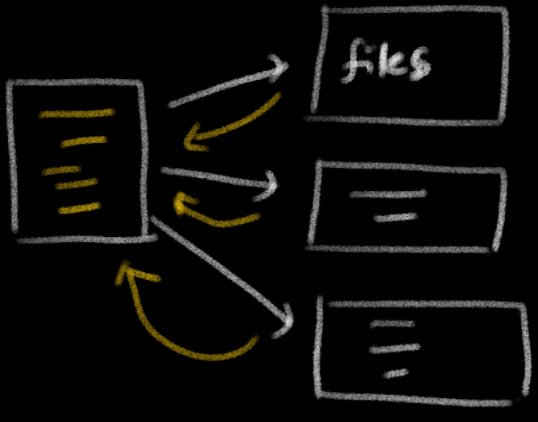


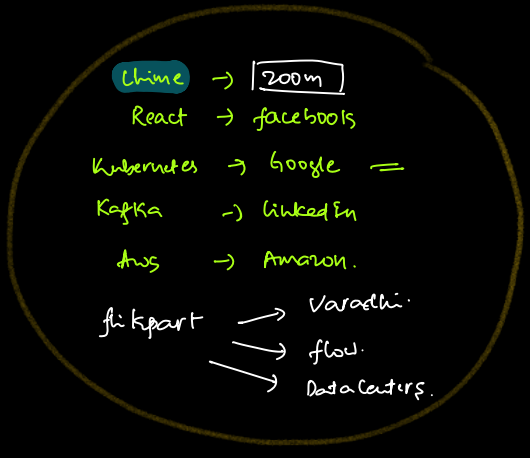
conflict.

D1 & D2 sit together & decide the final version.

ex: github.

Git → Linus Torvalds. [Linux].





commit

commit means save.

commit consists

1. commit id
2. message
3. timestamp
4. author
5. changes

Every commit consists of changes that are done after the last commit id.

Every commit is immutable

every commit is trackable.