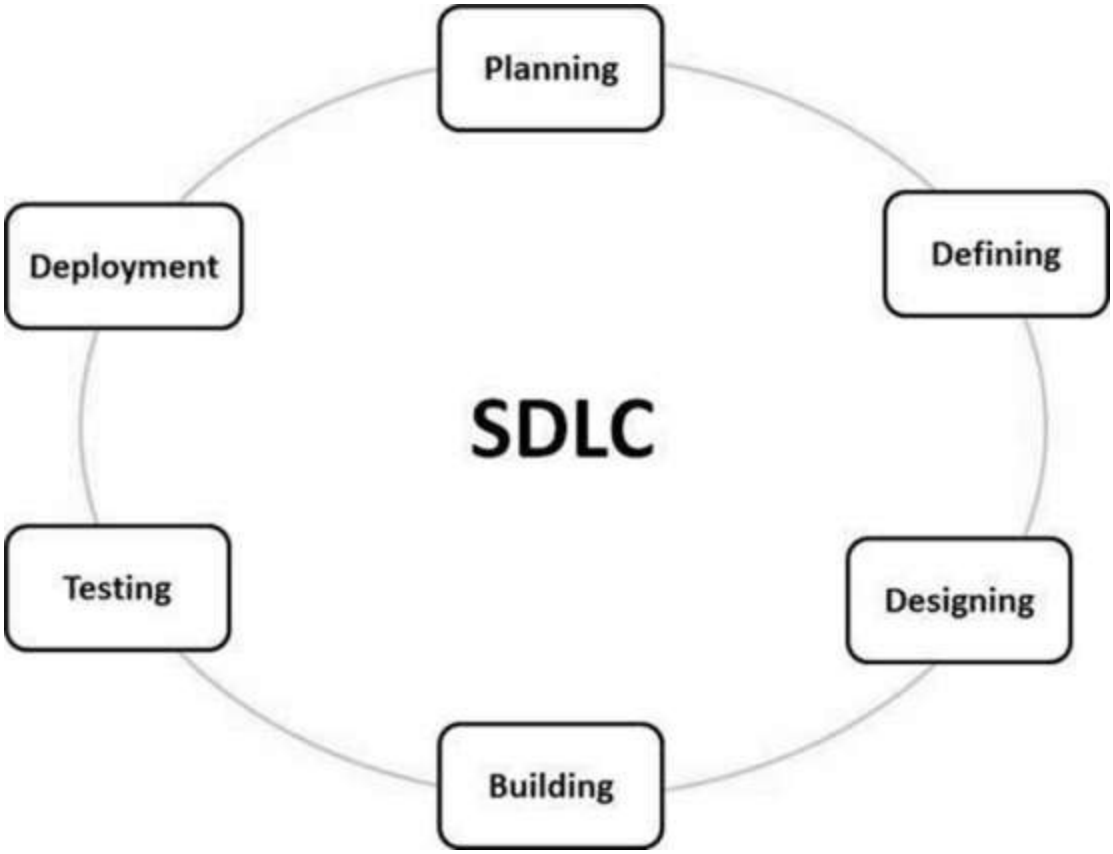
SDLC

A process used to design, develop and test high quality softwares.



Planning

Basically where you plan the application or development you want to do

Designing

Designing phase, where you discuss the details of how to implement the developments/features you want to have

Building

When you actually build out your application.

Testing

Unit testing - make sure the components of your code works

Integration testing - make sure the components of your code work well together

User Acceptance Testing - make sure your user experience is satisfactory

Deployment

When you actually deploy the feature/app into the public

Types

Big Bang

- Design?, build!, test?
- Usually used for prototypes
- Good for when you're trying to learn new languages or testing out a new feature

Waterfall

- Highly documented
- Each step is only gone through once
- Recommended for projects where design and planning are important and iterations are not really possible maybe because its expensive
- Like software for military projects
- Rigid, not flexible, you don't really need all that documentation for regular apps

V-Model

- AKA verification and validation model
- An extension of the waterfall model and is based on the association of a testing phase for each corresponding development stage
- the corresponding testing phase of the development phase is planned in parallel
- Highly tested application like software for rockets

Agile

- Iterative model
- Not a methodology, more of a guideline
- Like an interface but for SDLCs
- It's a manifesto
- Some Values:
    - People over processes

- Flexible
- Constant communication with everyone involved
    - Stakeholders and clients
- Break up work into user stories and work on each semi independently
- fast

Scrum

- Popular implementation of the agile methodology

Phases:

Sprint Planning

- When you take user stories and assign points to them
- Points are the estimated amount of effort it would take to accomplish a user story
- Usually limited with a capacity - the number of work hours you have

Sprint

- The time you actually work on your user stories
- You take a bunch of user stories and you work on them
- Usually lasts 2-6 weeks
- The whole team is on the same sprint
- Frequent scrum meetings
    - aka standup
    - Daily
    - 3 parts: what you've worked on, what you're planning to work on, any blockers
    - Stand up as an incentive to sit back down again
    - Led by a scrum master/team lead

Sprint Review

- Review the doneness of your increment
- Verify increment against the definition of done and acceptance criteria

Sprint Retrospective

- Review what happened in the last sprint
- Take a look at your burndown chart
- Evaluate your velocity - how many points finished in the last sprint

Artifacts:

Burndown chart

- Tracks your progress throughout the sprint

Project board

- Source of truth about the big picture of the development process
- Maintained by the scrum master

YOLO Development

- **Patent pending**
- Not an actual (recognized) development methodology
- Actual development process is like in big bang but it makes it into production
- Only has one iteration like in waterfall
- Not really documented
- Flexible(?) to changing requirements like agile
- All about taking requirements (if it exists) from product owner/ project manager and coming up with something to show on project presentation
    - Requirements are very open to interpretation
    - All about the developer showcasing their creativity by coming up with their own implementation following the requirements
- Most apps developed under YOLO development are made once, released, and never updated
- Start (Some sort of requirements) >>>>>>Start panicking>>>>>>>Panic (Rechecking the requirements, refactoring the code) >>>>>>>>>Panic some more>>>>>>>End (Some sort of MVP, released to production)