Mobile and Embedded Computing

Lecture 0. Course orientation

Ing. Dinu-Ștefan RUSU

You can contact me on:

- dinu_stefan.rusu@upb.ro
- Teams

Senior Engineer @ Dell [Kotlin, Spring Boot]

45+ Launched apps on Google Play & App Store [Flutter, Java]

Speaker at DevTalks 2025 – Scaling Flutter to 100k MAU as a solo dev

Course Topics

1. Git & GitHub, Web vs PWA vs Cross platform vs Native, Overview of SSR, SSG, CSR
2. Compiled vs Interpreted languages, Null safe languages, Dart & Flutter
3. Agent assisted coding, Async vs Threads, Flutter Widgets & UI Elements
4. Stateless vs Stateful, State management techniques (introduction)
5. Server vs Client-Side code execution, Serverless vs VPS (+ http & Firebase)
6. GraphQL & REST, Observability for Mobile applications (Events, Crashlytics)
7. Web Sockets, Permissions & routing
8. State management techniques, Business Logic Component (BLoC) pattern
9. Authentication, OAuth providers, Firebase App Check
10. AI On-device (Google ML Kit for Flutter) & Connecting to LLMs (VertexAI)
11. Deploy to stores (Google Play, App Store) & CI/CD With GitHub Actions
12. Performance and Energy Consumption

Grading

Final grade (10p)

4p (Exam)

+ 4p (Laboratory)

+ 2p (Course Test)

Requirements

In total ≥ 5

(Exam + Lab + Course Test)

Laboratory – 4P

Hands-on Flutter exercises

4 Milestones:

1. Build basic UI (0.5p)
2. Integrate with the provided API (0.5p)
3. State management & Offline first (1p)
4. Final presentation (2p)

Course test – 2P

Written theory, on paper

1h

During lecture hours

Final exam – 4P

Written theory, on paper

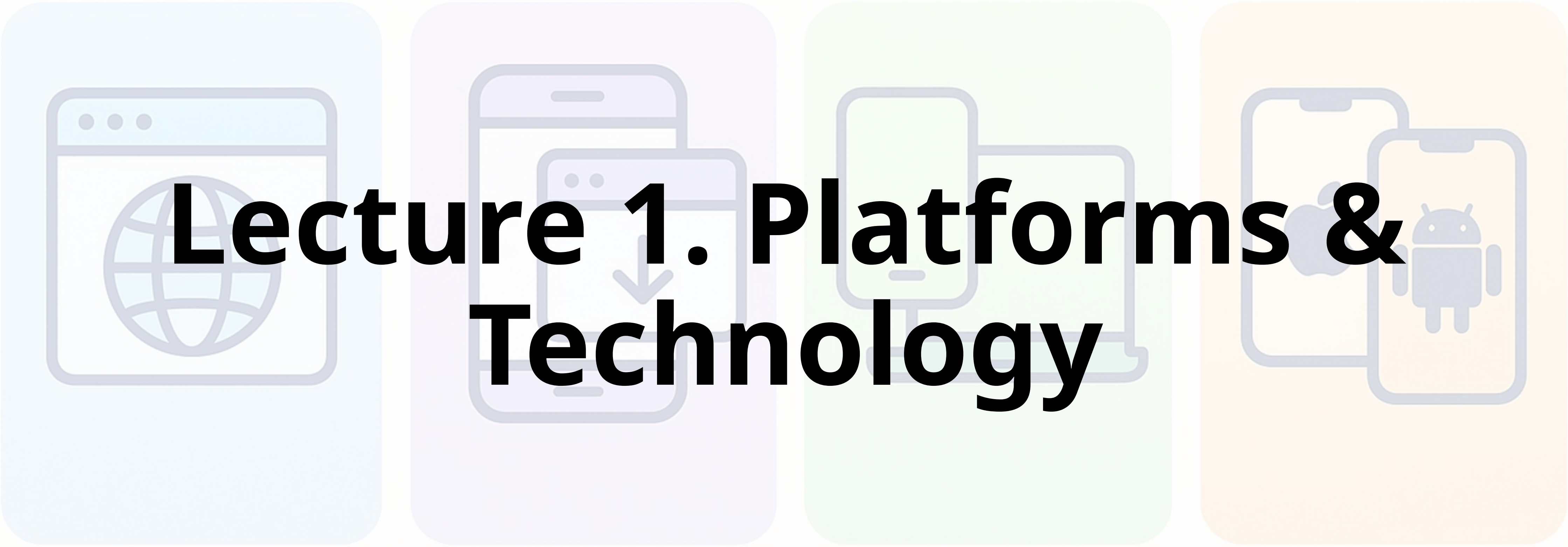
During exam session

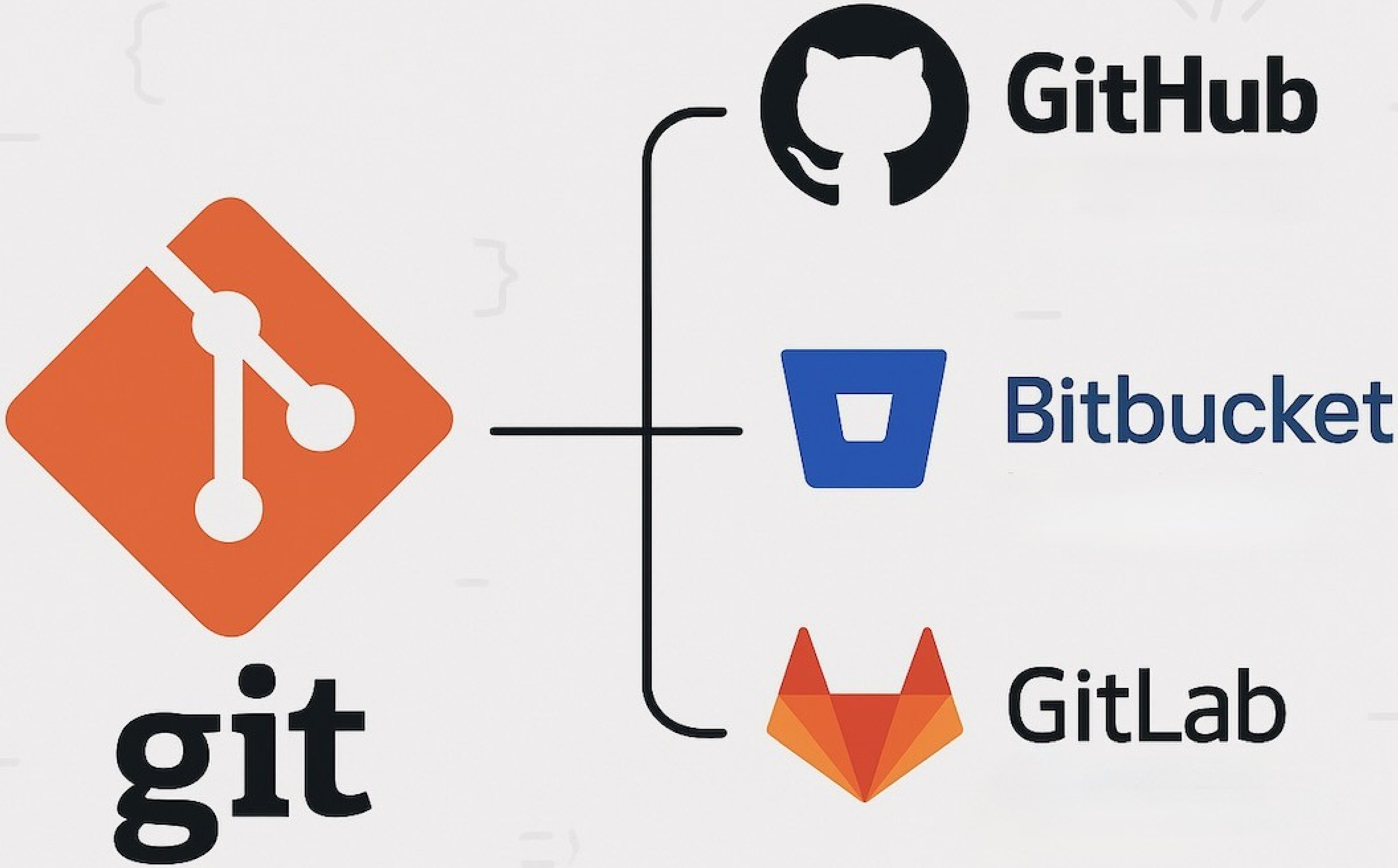
Bonus points

Kahoot

Course answers

Lecture 1. Platforms & Technology




git
GitHub
Bitbucket
GitLab

- Git is a version control system, designed to handle everything related to source code management

- Used to track changes in the source code, enabling multiple developers to work together

- It is mandatory to create a GitHub account at https://github.com/signup

- You can find all the resources you need in this repository: https://rusudinu/mobile-and-embedded-computing

- A repository contains all the files of your project, and each revision of it

- It contains at least one branch (main/master)

- A repository is a Git ‘concept’, and NOT a GitHub one

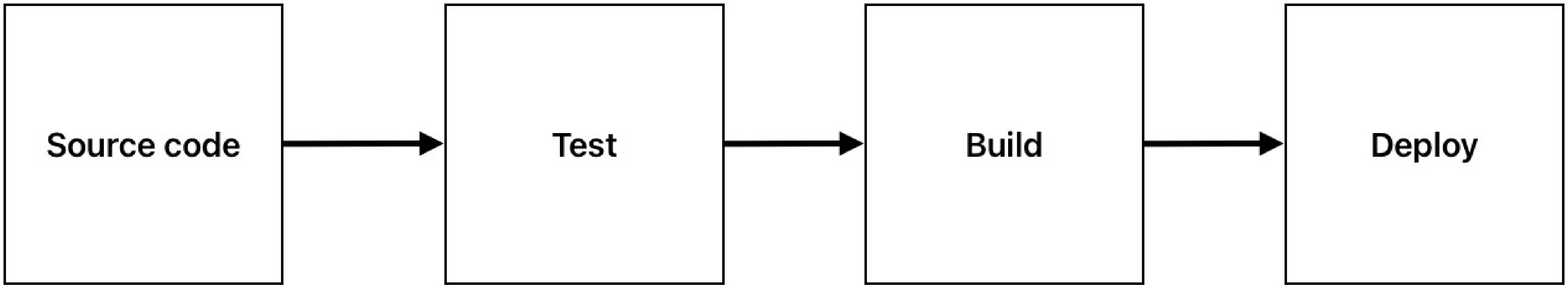
- Branches allow contributors to develop features, fix bugs or experiment new ideas without impacting other team members.

- A branch is created from an existing branch.

- A pull request is an event that takes place when a contributor is ready to begin the process of merging new code changes with the main project branch.

- After a peer review, a pull request may or may not be approved.

- If approved, it may be merged, otherwise the author must make the requested changes until he receives all the required approvals.



Source code

Code repo

Test

- Unit
- UI
- Integration

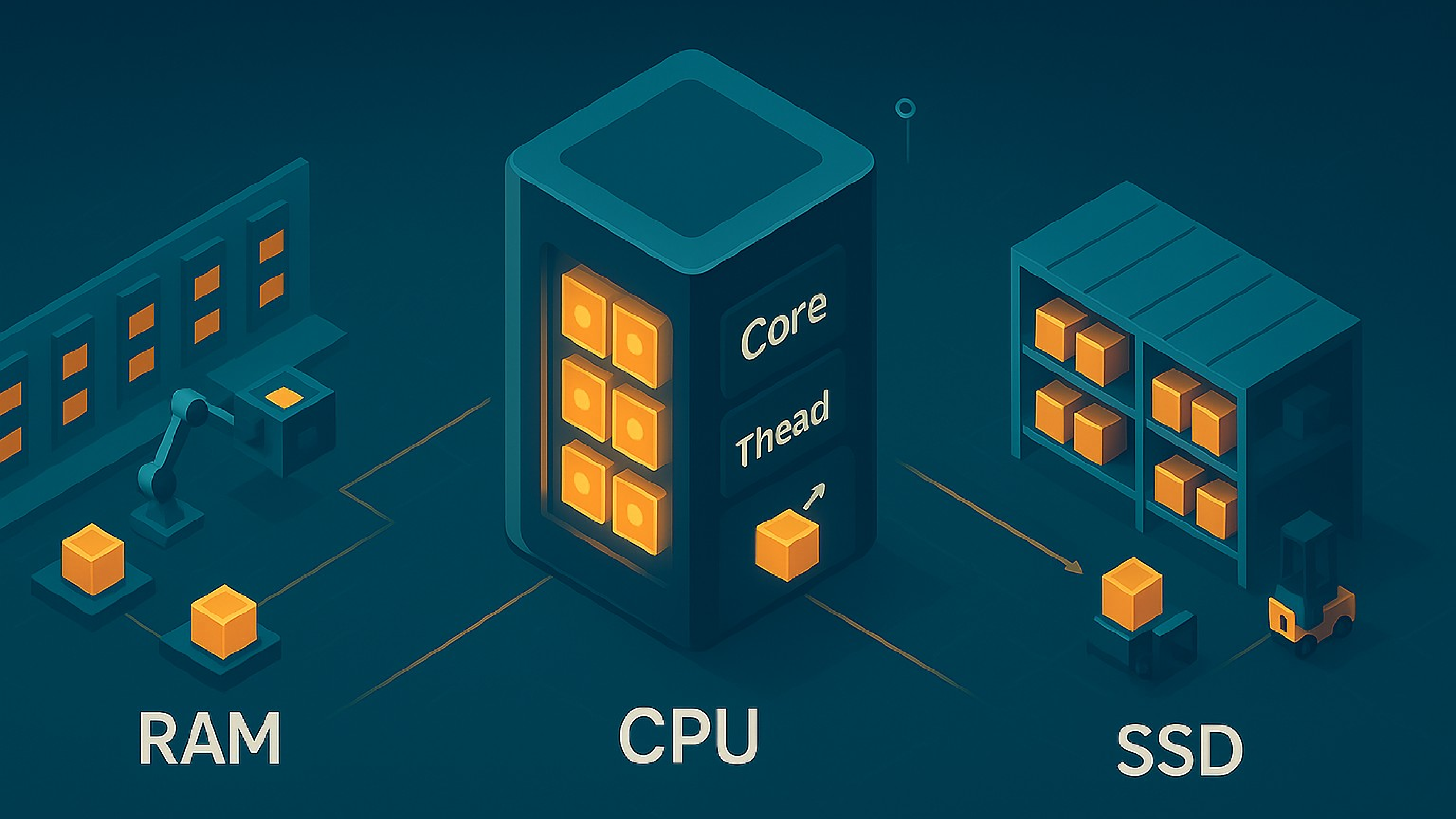
Build

- Code & dependencies
- Code signing

Deploy

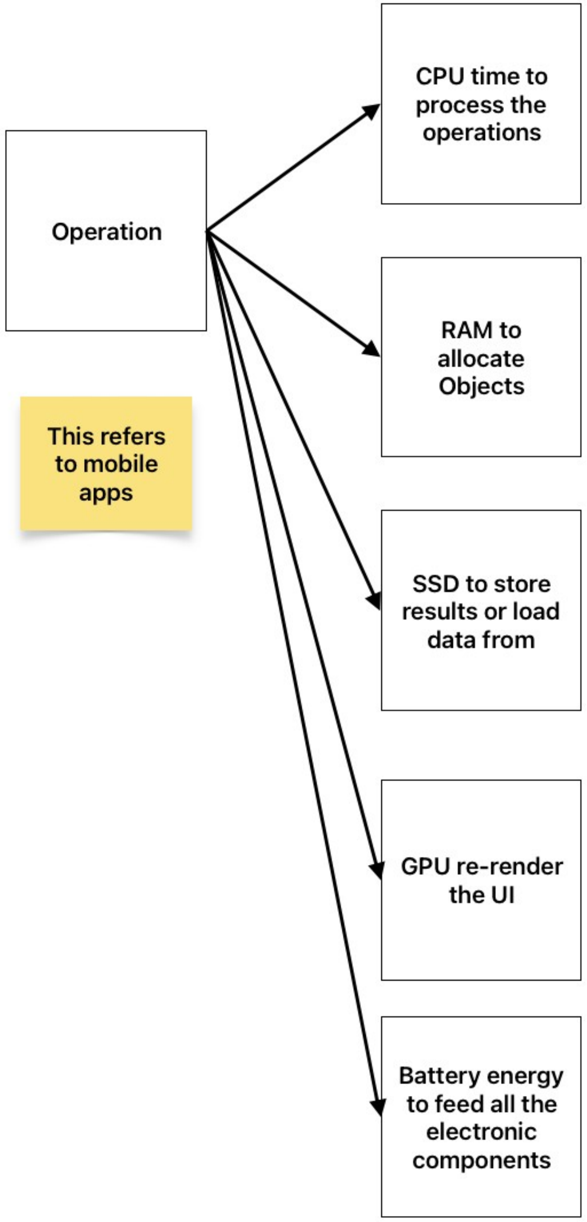
- Internal testing
- Alpha
- Beta
- Production

In production we also have the concept of 'Feature Flags'


Core
Thead
RAM
CPU
SSD

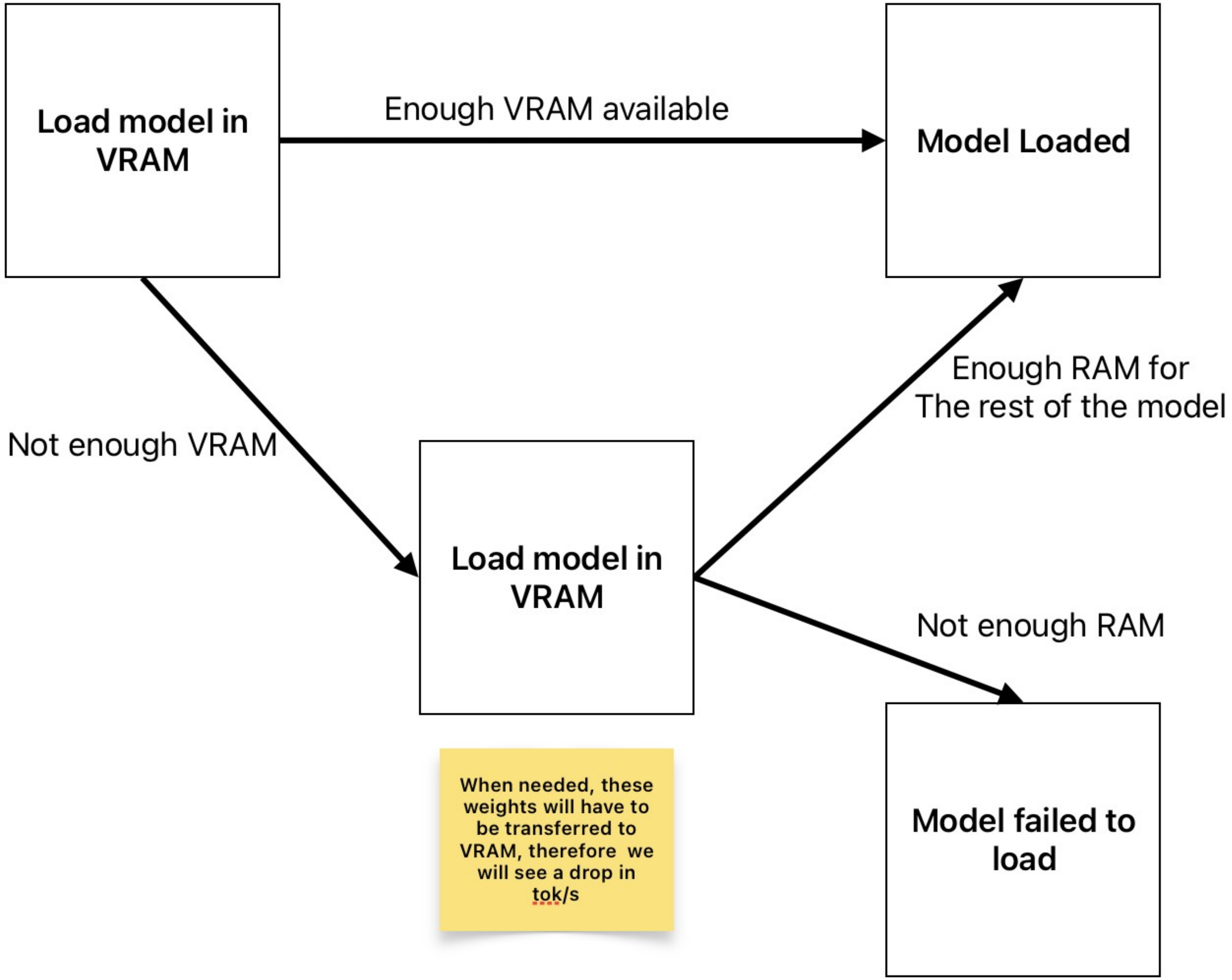
Device resources

- CPU
- RAM
- SSD
- GPU
- For mobile devices, the battery is also a 'resource'


Operation
This refers to mobile apps
CPU time to process the operations
RAM to allocate Objects
SSD to store results or load data from
GPU re-render the UI
Battery energy to feed all the electronic components

Resource allocation for LLMs

- When you try to load a Large Language Model (LLM), the operating system will do the following
- 'Unified memory' in MacBooks is very useful for LLMS because you have a lot more 'VRAM' available



When needed, these weights will have to be transferred to VRAM, therefore we will see a drop in tok/s

Web App

- Runs in a **web browser** instead of being installed directly on a device like a Desktop App / Mobile App
- Is accessed using a **URL** (e.g. https://rusudinu.ro)
- No installation needed
- Cross platform ‘by default’
- Usually require an active internet connection
- The content & functionality updates in real time
- ‘Plain’ HTML & Js, Angular, React, Vue, Svelte, etc

Progressive Web App (PWA)

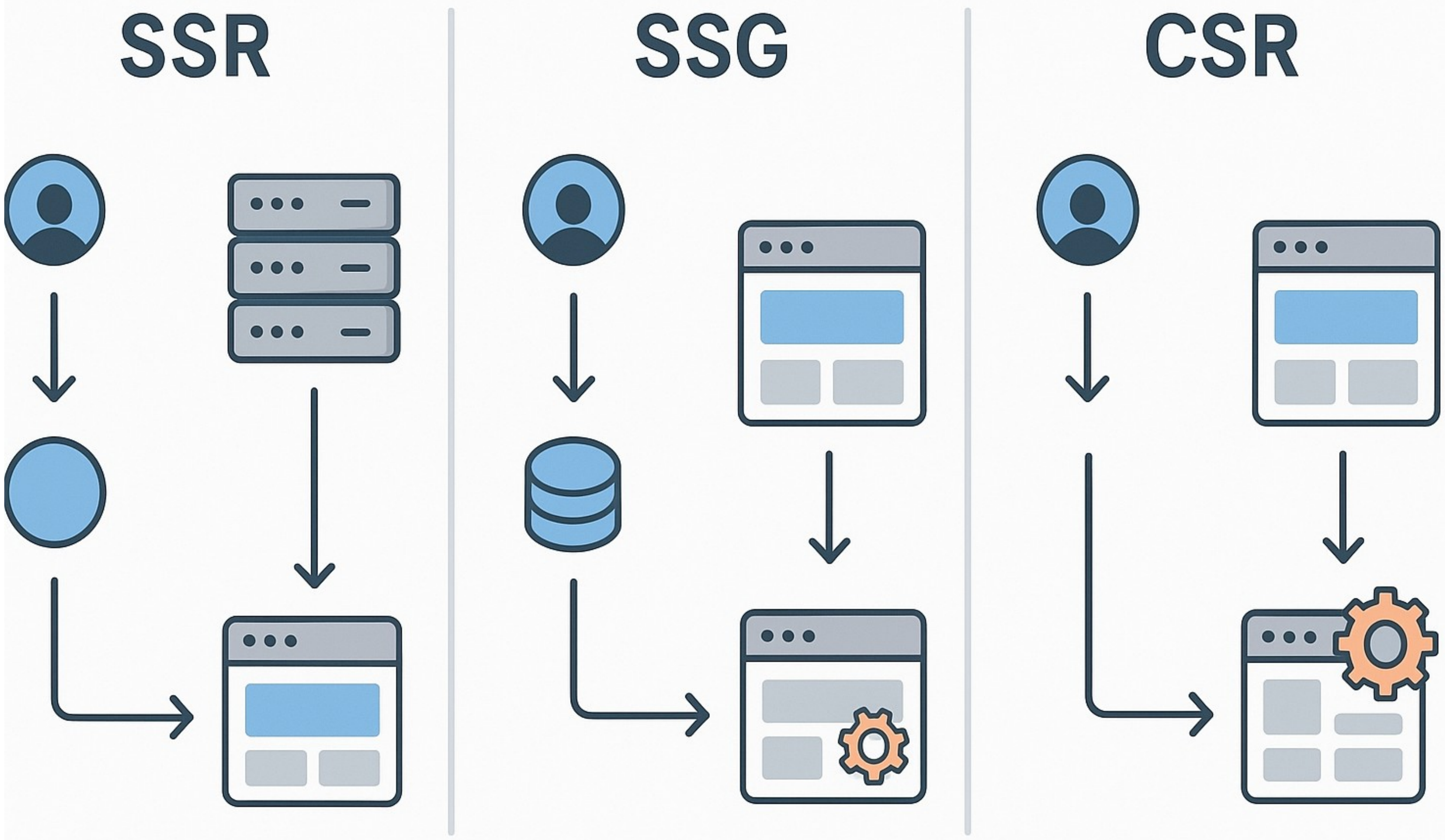
- ‘Modern’ technologies making a website behave like a mobile app
- Installable
- Capable of working offline
- HTTPs only
- Cross platform
- Updates are automatically checked in the background by the service worker and are applied when the app is opened the next time
- ‘Plain’ HTML & Js, Angular, React, Vue, Svelte, etc

Cross Platform Apps

- Most of the code is written once and the app will run on Android and iOS
- Better performance compared to PWA
- Slower to update, need to go through the Store review process (except for some cases like parts of React Native apps that can receive OTA (Over the Air) updates)
- Some have Desktop support (Flutter)
- Can work well fully offline
- Flutter, React Native, Kotlin Multiplatform, etc
- Published in Google Play, App Store

Native Apps

- Platform specific, built for one OS (iOS or Android)
  - iOS -> Swift / ObjC in Xcode
  - Android -> Kotlin / Java in Android Studio
- Can work well fully offline
- Best performance
- Use native components
- Published in Google Play, App Store



SSR
SSG
CSR

Server Side Rendering (SSR)

- HTML is generated on the server for each request
- Better SEO since content is immediately available to crawlers
- Faster initial page load and Time to First Contentful Paint (TTCP)
- Higher server load as each request requires processing
- Slower navigation between pages (full page reloads)
- Examples: Next.js with getServerSideProps, Nuxt.js, SvelteKit
- Ideal for dynamic content that changes frequently

Static Site Generation (SSG)

- ➢HTML pages are pre-built at build time
- ➢Excellent performance with fast loading times
- ➢Great SEO as content is fully rendered in HTML
- ➢Can be served from CDNs for global distribution
- ➢Limited to content known at build time
- ➢Requires rebuilding and redeploying for content updates
- ➢Examples: Next.js with getStaticProps, Gatsby, Astro, Jekyll
- ➢Used for documentation

Client Side Rendering (CSR)

- JavaScript runs in the browser to generate the page content
- Smooth user experience with fast navigation after initial load
- Rich interactivity and dynamic user interfaces
- Slower initial page load as JS bundle must be downloaded and executed
- Poor SEO without additional optimization (crawlers see empty page initially)
- Examples: Create React App, Vue CLI apps, traditional SPAs
- Best for highly interactive applications and dashboards