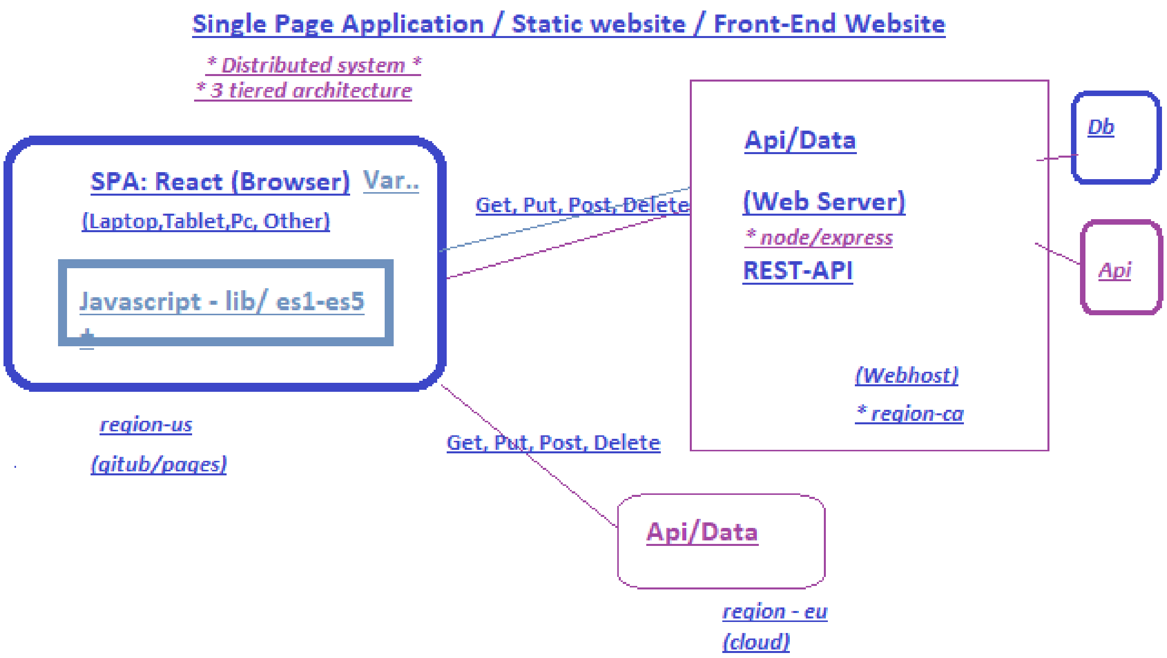
Agile, Modular, Open source development - poc - prototyping, refactor, best practices ie: Git , SDLC



Dynamic js website development
Dynamically js updating webpage without refreshing the entire web page
- website data
- website scale to 1000, mill user request

Front End- UI/UX

Websites, Web Applications,
HTML5, CSS, SEO
SPA - Single page websites
SAAS - Software as a Service - online web applications ie: Ms office online, GitHub, cloud systems etc...
Pwa - web/mobile hybrid
Other

Modular agile web development
Responsive Ui/ux - client, stateless
Web design prototype
Reusable web components
or copy components into webpage

SPA- React - No full web page refresh
- client state data
- no full page refresh

REST API data request, response
Representational State Transfer - Restful API
A MVC stateless protocol used to Retrieve, Insert, Update and Delete data using rout verbs such as GET, POST, PUT, DELETE
- latency, errors, timeout, reprocessing
- multiple requests, errors, reprocessing
- agitating multiple request - user list, set, map / redux
- Pagination, request limiting, throttling, blocking, async await

----

client request architecture

----

thousands to mil request / response
-> browser (download js,css, SPA-react) -> server request/response
-> browser (api-request) -> server request/response

Client
** thousands to mil request / response \*
(Browser-js-es1...5+)-region
(Tablet-js-es1...5+)-region
(Mobile-js-es1...5+)-region

[Server] - region ie: CA
* thousands to mil request / response

React / DOM / SPA - no browser refresh

Backend server development

Web server - Request, Response
MVC - model view controller,
 - rout or path for each request
 - Data, session, stateful
 - create client html SEO
SQL - rdbms, MySQL, postgresql, other
NOSQL - mongodb
- latency, errors, timeout
- multiple requests, errors
- client request, response

....

3 tiered / 3 layered architecture
MVC - Rest API -sql/nosql- webpages and SPA
- handle more quest/response, more scalable
- decoupled architecture, more failure points and unit testing

2 tiered / 2 layered architecture
MVC - SQL/NOSQL - webpages and SPA
- handle lest quest/response, less scalable
- coupled architecture, less failure points and unit testing

Build javascript front-end and backend websites

NODE - npm, express

REST API - middleware

- client request, response
- latency, errors, timeout
- multiple requests, errors