**High Level View - Minimalistic**

**Hardware Side:**

HL Design is just a solution to a problem.

-If you said you need a car, the solution is that you get a car.

-It doesn't matter the process, just that you get the car.

-What is the car?

-What is the specific solution to implement?

**Front End**

-Whatever is running the code that the user interacts with.

-Application running program:

-Browser

-Device

-Desktop

-Front End talks to Web Server

**Back End**

-Web Server - executes complex logic - implementer of the feature itself

-Talks to DB

-Database - SQL or relational - Not specifying type at this level

-Relational Database = SQL Database

-Relational Database Management System (RDBMS) - same as above, but

specifically what program

-Oracle

-MariaDB

-MySQL

-Database Server

-Seperate than Web Server

-Runs on own server

**Software Side:**

-Web

-PWA - Service Worker (Becoming the standard - goal for most web apps)

-Behaves like a mobile application

-Saves interactions offline and sends information once

connected again

-Mobile

-Desktop

-Don’t use monolithic

M - Model - Class

-Car. Description of entity. No logic.

-Door

-Color

-Engine

-Transaction

-Price

-Date

-Person

V - View - UI

-HTML

-Koflin/Jason

-Swift

-C++

C - Controller - Class - Routing/Navigation or Satisfy Request

-If you need a feature or to satisfy a request, the controller will satisfy

by updating DB or whatever needs to be done.

-Lives on Web Server

VM - View Model - Class - Runs on Client Side (Front End)

-Lives on UI

-Hanldes any logic needed for that specific view

-What happens when you click this button? (for specific view)

SPA - Single Page Application - Doesn't use service worker

- each new page = new html file

PWA - Progressive Web Application - Uses service worker

-Most applications use a SPA with an MVVM or MVVC

-Synchronous HTTP request for initial request, returns

1 html file

-Future requests are asynchronous - will update

the 1 html file dynamically

Cannot skip flows in Architecture

-May need to consider flow for when using or not using the service worker (SQ)

**Deciding on Architecture (High Level - Minimalistic):**

-Using MVVM

-User -> View -> VM -> Back End

-Using SPA

-User -> Initial HTML - Client side - Back End

-SPA with MVVM

-HTML -> VM -> Back End

-Front End

-View -> VM -> ...

-Back End

... Web Server (Handler/Controller) -> SQL DB

**Design choices:**

-Where to handle input validation

-Where to handle Error Handling

-Where to handle Logging

-Where to handle Security

-At specific points in FE or BE?

-At all points?

-Designing a system that's agnostic to technology, we can

say that as long as there is an acceptable replacement

for a tech, our application will continue to work

-Keep generic

-Specific technology limits app to that tech

-Need to consider future features as well when considering

design choices - future proofing