# Oppgave 1 – Bookingløsning

## 1a) Typer/roller for en booking system:

* Administrator
  + Egen webside for å overvåke aktiviteter
  + Gi rettigheter og definere roller
  + Opprette brukere
  + Rydde opp i feil registreringer
  + Gi assistanse til kundebehandlere ved behov
  + Definere innhold og tilbud for kundetyper
  + Tilgjengeliggjøre squashbaner
* Kundebehandlere
  + Se kundens fakturahistorikk
  + Se kundens aktivitetslogg
  + Registrere kunder
  + Avslutte kundeforhold
  + Lese/skrive kundenotat
  + Kommunisere med kunder
  + Hjelpe kunder når de ikke får til å bruke systemet
  + Reserveringsoversikt og administrasjon
  + Registrere mangler
* Kunde (Her kan det være flere kundetyper avhengig av abonnementstyper)
  + Behandle bestilling
  + Innhente fakturainformasjon
  + Aktivitetslogg
  + Registrere mangler
  + Angi kundetilfredshet
  + Kommunisere med kundebehandlere
* Vedlikeholds personell
  + De trenger å planlegge når de skal vaske/fikse noe
  + reservere squashbaner
  + oppgi status på squashbaner
  + Oversikt over rapporterte mangler
* Personlige trenere
  + Se tildelte kundens aktivitetslogg
  + Se bookinginformasjon
  + Reservere banker
* Project manager\*
  + Hente fakturahistorikk
* Reservasjonssystemet:
  + Arkivere gammelt informasjon
  + Registrere aktivitetslogg.

## 1b) Beskrivelse av løsning av systemet:

Her er det vanskelig å påstå at en plattform og teknologi er bedre enn noe annet:

* Et reservasjonssystem for et treningssystem må være sikkert, men ikke på samme nivå som for eksempel en bank. Det er dermed lavt behov for overvåkning av systemsikkerheten.
* Det er heller ikke behov for stort søkelys på stabilitet. Konsekvensen av at systemet er nede er lav. Funksjonaliteten kan erstattes med et papirsystem for kortere periode og eventuell tap av informasjon er trist, men ikke kritisk. Det er dermed også lite behov for overvåkning av systemressurser.
* Det kommer antageligvis ikke være stor trykk på systemet heller da vi snakker om et treningssenter, så robusthet er heller ikke en prioritering.
* Det er heller ikke stort behov for at det skal være forferdelig raskt. Det trenger å svare før kunden blir lei, men trenger ikke å håndtere millioner av transaksjoner i minutter.
* Det er ikke avanserte oppgaver som skal gjøres i systemet så her trenger man ikke en utpreget brukerinteraksjon, men selvsagt så må selve reservering av systemet presenteres veldig rent og enkelt.
* Det er heller ikke nevnt om dette skal utvikles av meg alene eller sammen med andre. Når man jobber sammen med andre så må man gjøre vurdering av teknologier i lys av det.
  + Hvilke sett med teknologi er alle komfortabelt med?
  + Verktøyer for å organisere stories og oppgaver?
  + GIT -kjøreregler?
  + Kodestandard?
* Hensyn må tas til grafikk og utseende da treningssenterets kunder skal bruke det.

Dette er dermed en oppgave som kan stort sett løses av alle plattformer og teknologier. Her kan man godt ha en PHP-wordpress løsning på webhotell med MariaSQL og kunden vil være fornøyd. Det finnes nok allerede. Mitt valg ville ha vært basert på at behovene for treningssenteret utvikler etter hvert, og eventuelt behov for integrasjon mot andre systemer:

* Java+Spring
  + Kjapt å sette i gang med (start.spring.io)
  + Mulig legge til flere funksjonaliteter hvis behov
  + God støtte
* Git
  + Standard verktøy for versjonering av koden
* Javascript- React
  + Front-end magi
  + Lettere å organisere koden i forhold til Angular
  + Virtua DOM
* Javascript -webpack
  + JS build tools
* Maven
  + Bygging og deployment av koden
  + Dependency management
* Visual code
* NPM
  + JS package manger
* Azure/AWS/google for deployment.
  + No on-premise solution
  + Standard of the day



## 1c) Nye muligheter for ledelsen:

* Utvikle spisset tilbud ut ifra aktivitetslogg
* Analyse
  + Hvilke baner som er mest populært?
  + Hvilke tidspunkt som er mest populært?
  + Hvor ofte er en bane brukt før vedlikehold?
  + Personell historikk
* Fakturaoversikt
* Effektivitetsøkning

# Oppgave 2: Dårlig Ytelse

[…]”av og til går alt tregt, men noen ganger går det raskt.”[…]

2a) Elementer som kan spille inn:

Sortert etter sannsynlighet:

* Nettverk (proxy, dårlig DNS)
  + Måle Ping fra kundes nettverk til lastbalanser over tid (DNS VS IP)
* Database
  + Deadlocks: Avhengig av database. De fleste har noe query du kan kjøre mot systemtabeller
  + For mange connections ->Dead connections: query systemtabeller for inactive, men tilkoblet tilkoblinger
* Lastbalanser algoritme
  + Lastbalanser konfig: Sjekke hvilken algoritme den bruker (round robin, etc)
* Applikasjonsserver ressurs
  + Antall tilkoblinger per webserver (bør være likt, men av og til så er det noe connection som «henger»): Netstat kommandoen. Noen applikasjonsservere har metric som det er mulig å få tilgang til.
  + Heap memory problemer (koden ikke optimalisert for garbage collection strategi?) - JvmTop
  + Hibernate query på en admin funksjon som henter opp hele databasen?: Analyse av hibernate relation mapping og logging.

NB! Før jeg hadde begynt å analysere så jeg restartet applikasjonsservere og sett om det hadde løst noe, men jeg ville ha begynt med antakelsen at nettverk eller database som er problemet.

# Oppgave 3 Heiskontroll

## 3b) Kontrollsystem utfordringer

Et kontrollsystem for en heis skal være stabil og sikker. Ingen skal fanges i en heis på grunn av feil i programvaren. Programvaren skal også ta hensyn til at forutsetningen er til stede for at heisen skal fungere feilfritt. Den skal så godt som den kan forutse vedlikeholdsbehov og sørge for heisen kjøres så effektiv som den kan gjøre: Altså minimiserer tiden for den blir tilgjengelig. Tilleggsfunksjonalitet for å overstyre etasjerekkefølgen bør implementeres.

Funksjonelle utfordringer:

* Behagelig opplevelse (Unngå plutselig akselerasjon).
* Vekt
* Antall personer
* Dørsensorene er aktivisert for lenge
* Mekanisk slitasje
* Brann
* Ambulanse
* Andre katastrofer
* Rettferdighet

Programvare for en heiskontroll skal måles etter stabilitet.

Tekniske utfordringer:

* Redundans
* Stabilitet
* Sikkerhet
* Gain
* Sensortolkning
* Akselerasjon
* Vedlikehold