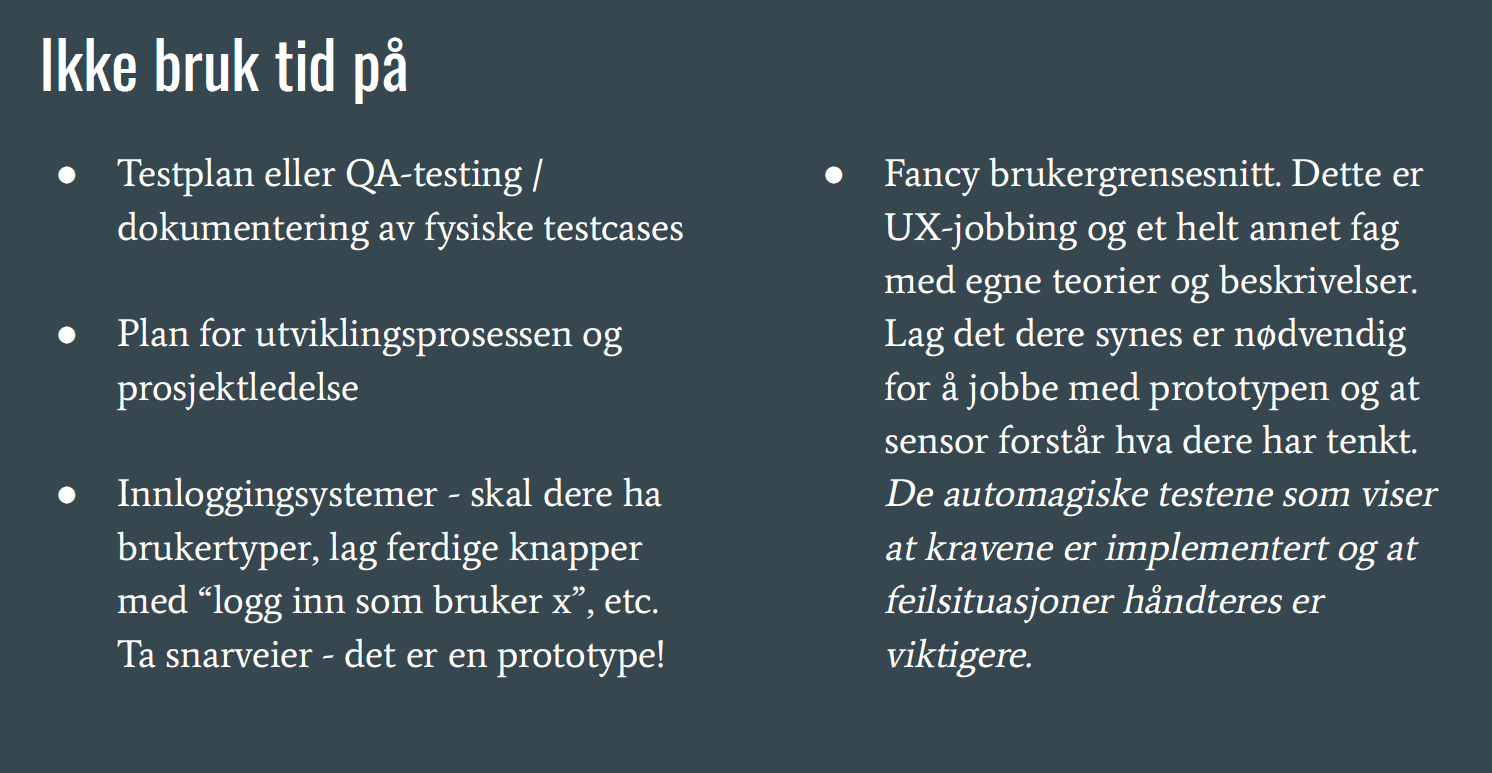
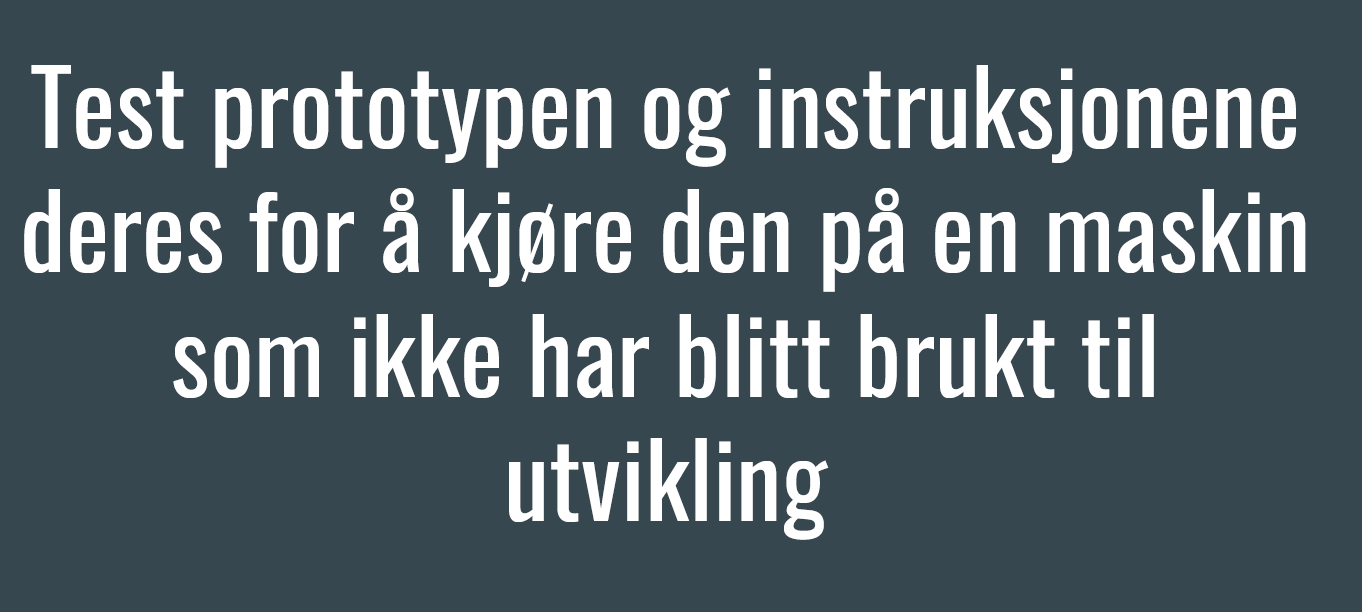
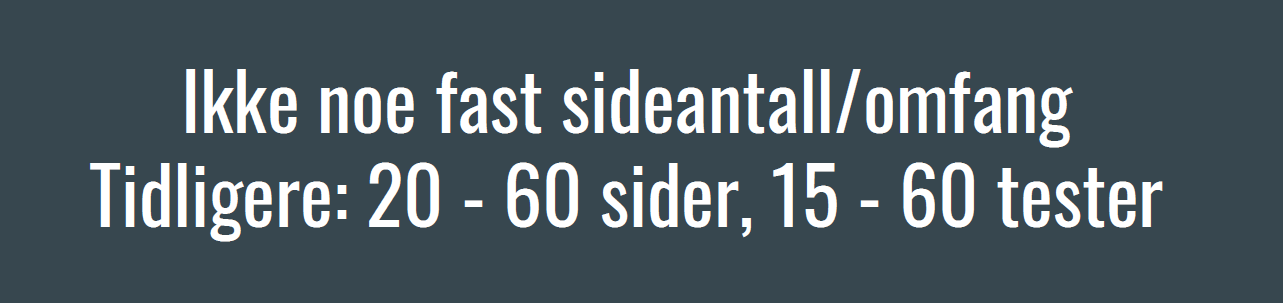
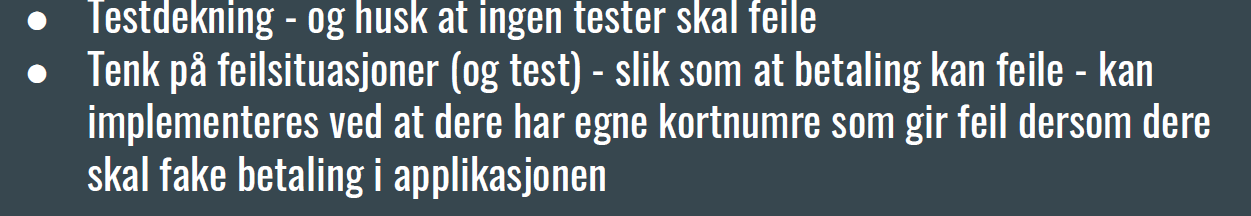
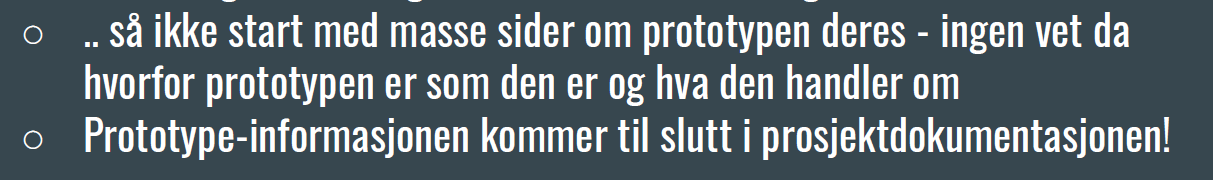
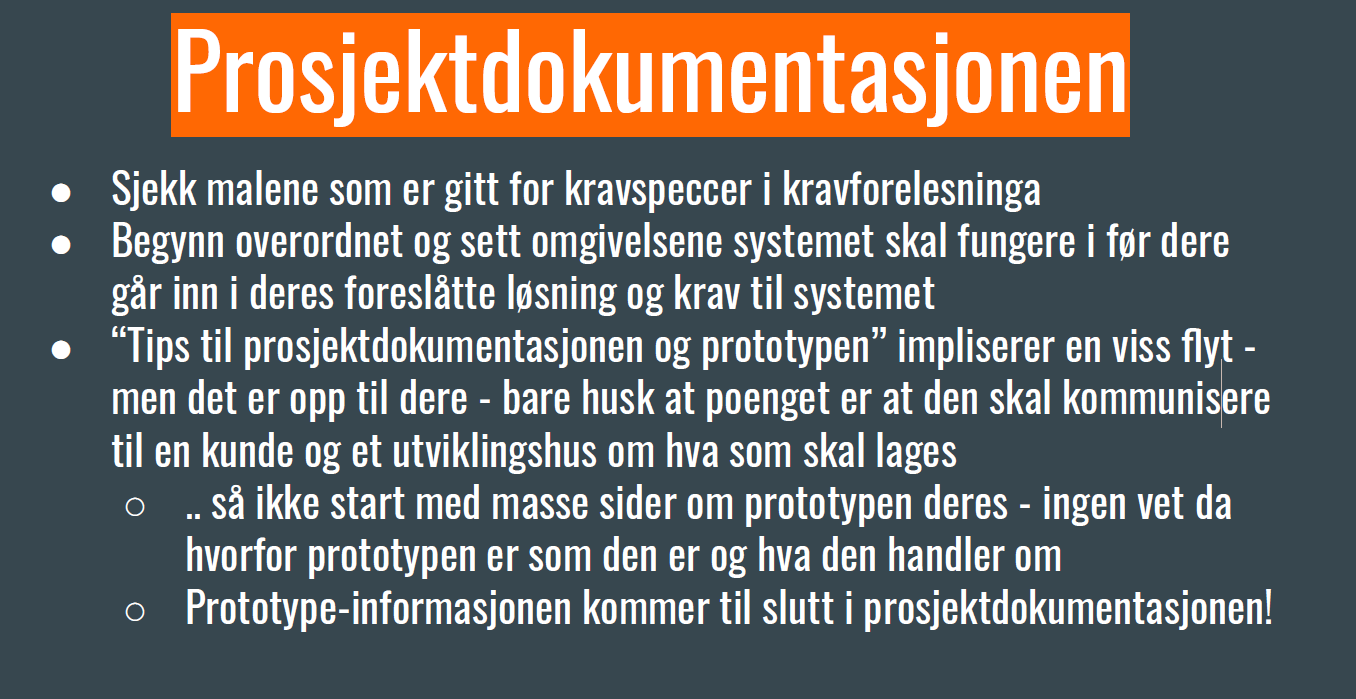
Dokumentasjon plan

* Beskriv begrensninger og antagelser dere har gjort i prosjektperioden
* Introduser brukerne av systemet (gjerne beskrevet som såkalte “Personas”) og hvilke oppgaver de ønsker å løse i systemet (f.eks. “Use cases”). Utbrodér de forskjellige brukssituasjonene i mer detaljerte beskrivelser, slik at det er mulig for utenforstående å forstå hva oppgavene som brukerne skal løse i systemet går ut på.
* Introduser brukerne av systemet (gjerne beskrevet som såkalte “Personas”) og hvilke oppgaver de ønsker å løse i systemet (f.eks. “Use cases”).
* .En overordnet modell av prototypen og en beskrivelse av hvordan delene i den hengersammen vil være nyttig for å illustrere hvordan prototypen fungerer. Beskrivelsen skal også vise hvordan prototypen henger sammen med det større systemet dere beskriver.
* Forklar hva som må til for at prototypen deres kan kjøres, og hva en bruker ev. må konfigurere før den kjører.
* Forklar ev. kjente svakheter og problemer med prototypen. Det er bedre at Kunden vet om problemer på forhånd før de ev. merker dem selv.
* gjennom dokumentasjonen og forstå problemet, hva som skal utvikles og hva som er viktig med systemet som utvikles
* Hver person på gruppa skal lage to diagrammer – Disse diagrammene kan være enten dataflytdiagram, sekvensdiagram, tilstandsdiagram eller aktivitetsdiagram. Diagrammene skal vise ulike funksjoner i systemet. Legg inn diagrammene der det er relevant i teksten, ikke for seg selv ved siden av eller til slutt i prosjektdokumentasjonen
* Beskriv gjerne personas og utvidede brukerhistorier i prosjektdokumentasjonen!
* Notér hvilke krav som er implementert i prototypen i prosjektdokumentasjonen - f.eks. ved å bare notere [implementert] på kravene.
  + Lenk kravene til hvilken test som viser at kravet er implementert
  + Denne krav <-> test-relasjonen viser at dere tester viktig funksjonalitet og sørger for at dere passer på testene underveis!
* Lenk gjerne testene til kravene også - // Implements PARKING.PAYMENT.01
* Gruppér kravene etter funksjonsområde - ikke etter bruker
* Antall tester? 15 - 150
* Det første sensor gjerne gjør er å kjøre testene - pass på at INGEN feiler i det det dere leverer (tester skal IKKE feile - om dere forventer feil resultat tilbake, så skal dere sammenligne med det gale resultatet) Testdekning - og husk at ingen tester skal feile
* “Run all tests” i IDE-et hvis ferdig oppsatt - hvis ikke, beskriv i prosjektdokumentasjonen og README.md hvordan testene kjøres!
* Tenk på feilsituasjoner (og test) - slik som at betaling kan feile – kan implementeres ved at dere har egne kortnumre som gir feil dersom dere skal fake betaling i applikasjonen
* Pass på at dere dokumenterer eksterne avhengigheter og avgrenser hva systemet skal ha ansvar for (og hva som håndteres av eksterne partnere).
* Det skal også lages nødvendige tester som viser at prototypen tilfredsstiller de viktigste kravene som dere har avdekket og som blir beskrevet i dokumentasjonen dere skal produsere for systemet.
* Det er viktig at dere passer på å dokumentere eksterne avhengigheter i systemet, dvs. der systemet må prate med systemer fra andre tjenesteleverandører (f.eks. betalingsløsning). Dere skal ikke integrere mot noen eksterne leverandører i prototypen, men bør skrive små “stubs” som gir dere muligheten til å teste funksjoner som avhenger av dem.
* Husk git mappa i innlevering
* Sørg for at det er beskrevet hvordan en person uten dyptgående IT-kunnskap kan bygge, kjøre og teste prototypen deres. Bør være “åpne prosjektet i IDE-et, trykk kjør”.
* Avhengigheter i applikasjonen deres skal kunne installeres automatisk ved hjelp av et pakkesystem for språket eller rammeverket dere.
* 
* 
* 
* Husk at prosjektdokumentasjonen skal kommunisere med folk som IKKE er IT-folk eller utviklere også!
* Sørg for å forklare problemstillingen, aktørene, og hva det foreslåtte systemet deres inneholder! (moduler, systemer,eksterne avhengigheter, etc.)
* 
* 
* 
* Det skal leveres en prosjektdokumentasjon som forklarer problemstillingen og løsningen dere kommer fram til, slik at Kunden og eksterne partnere kan lese gjennom dokumentasjonen og forstå problemet, hva som skal utvikles og hva som er viktig med systemet som utvikles. Husk at dokumentasjonen skal være forståelig for personer uten inngående kunnskap om fagområdet eller programmering.
* Kravene til systemet og systemet som helhet skal være dokumentert på et slikt nivå at dokumentasjonen kan sendes til eksterne selskap som kan utvikle, teste og evaluere systemet i samarbeid med produkteier og oppdragsgiver. Hvert krav skal inneholde gruppens eget estimat for utviklingsomfang og forretningsnytte i henhold til “T-Shirt sizing”-estimeringsmetoden (Small/Medium/Large/X-Large).

Prosjektdokumentasjonen skal være skrevet slik at utenforstående kan lese gjennom den og forstå problemet, hva som skal utvikles og hva som er viktig med systemet som utvikles. Husk at de som skal lese dokumentasjonen ikke nødvendigvis kjenner problemstillingen eller prosjektgruppens forslag til løsning fra tidligere.

* Beskriv problemstillingen og domenet for prosjektet som skal lages. Det er viktig at alle involverte har samme forståelse av hvilket problem som skal løses og i hvilke omgivelser programvareprosjektet skal brukes.
* Beskriv overordnet hva løsningen til prosjektgruppen går ut på og hvordan løsningen håndterer de forskjellige elementene i problemstillingen.
* Beskriv begrensninger og antagelser dere har gjort i prosjektperioden, slik at det er eksplisitt hvilke rammer prosjektet jobber innenfor. Husk på at vesentlige begrensninger og antagelser bør avklares med fagansvarlig.
* Introduser brukerne av systemet (gjerne beskrevet som såkalte “Personas”) og hvilke oppgaver de ønsker å løse i systemet (f.eks. “Use cases”). Utbrodér de forskjellige brukssituasjonene i mer detaljerte beskrivelser, slik at det er mulig for utenforstående å forstå hva oppgavene som brukerne skal løse i systemet går ut på.
* Kravene bør spore hvor i prototypen de er testet (f.eks. med ved å referere til testnavnet og filnavnet som testen ligger i), slik at det er enkelt å ha oversikt over hva prototypen inneholder og som er implementert.
* Husk forklaringer til tabeller og diagrammer
* Det er svært sjelden nødvendig med en fullstendig oversikt over systemet på detaljnivået i et klassediagram, men viktige og kompliserte deler av systemet bør beskrives overordnet slik at det er tydelig hvordan systemet er tenkt til å fungere.
* En overordnet modell av prototypen og en beskrivelse av hvordan delene i den henger sammen vil være nyttig for å illustrere hvordan prototypen fungerer. Beskrivelsen skal også vise hvordan prototypen henger sammen med det større systemet dere beskriver.
* Vis frem ting fra prototypen slik at Kunden får en forståelse for hvordan problemstillingen er løst, uten at de må kjøre prototypen på egen maskin (f.eks. med skjermbilder og beskrivelser). Dette er ikke en sluttbrukermanual, men en forklaring til hvordan dere har prototypet de viktigste kravene.
* Forklar hva som må til for at prototypen deres kan kjøres, og hva en bruker ev. må konfigurere før den kjører. Test instruksjonene på en maskin som ikke er brukt til å utvikle prosjektet. Husk at dette også gjelder hvordan testene til prototypen kan kjøres.
* Forklar ev. kjente svakheter og problemer med prototypen. Det er bedre at Kunden vet om problemer på forhånd før de ev. merker dem selv.
* Den som leser dokumentasjonen har ikke samme domenekunnskap som prosjektgruppa besitter etter å ha jobbet med problemstillingen. Sørg for å forklare nødvendige elementer for å unngå tvetydighet og misforståelser.
* Henvis til kravene som er nevnt i prosjektdokumentasjonen fra testene deres (“/\* Tester krav …. \*/”) og motsatt (i prosjektdokumentasjonen: “Testes i xyz/foo.java”)
* Pass på at testene deres faktisk tester funksjonene! Dersom dere endrer ting inne i funksjonene som testes på en slik måte at resultatet blir galt, så skal testen brekke. Det hjelper ikke om testen alltid passerer (“blir grønn”) - da er det ikke en test.
* Preface
* Introduction
* Glossary
* User reqirements defenition
* System architecture
* System models
* System evolution
* Appendincies
* index