Krav til innlevering

- ➔ Gruppeinnlevering bestående av 2-4 studenter
- ➔ Innlevering er 17.mars - 28. mars
- ➔ **Nett-studenter:** Oppgaven skal inneholde en videobesvarelse. Alle gruppemedlemmer skal snakke og presentere i videoen. Alle må ha på video og være synlige under presentasjonen. Videoen skal ikke overgå 8 minutter. Videobesvarelser over 8 minutter vil ikke bli vurdert.
- ➔ **Campus studenter**: Ta kontakt med Abebe for informasjon om muntlig fremføring av Arbeidskrav 2.
- ➔ Oppgaven må være et git-prosjekt og lastes opp til Github. Legg til foreleseren deres som collaborator (Nett - Amina: https://github.com/AminaBre, Campus - Abebe: https://github.com/abebediye)
- ➔ Repositoryets navn skal være: "**Arbeidskrav2-gruppenummer**".
- ➔ Alle gruppemedlemmer må ha commitet og vist lik innsats i Github. Dette betyr at ingen elever kan ta på seg mye mer eller mindre ansvar enn andre. Ikke slett noen brancher eller commits, all historikk skal være synlig.
- ➔ **Wiseflow-innlevering**
  - ◆ Dere skal levere en txt eller word-fil som inneholder:
    - Oversikt over github-brukernavnene deres + fullt navn
    - Lenke til deres Github-repository.
  - ◆ Selve oppgavemappen med kodebesvarelsen skal leveres som en zippet mappe.
  - ◆ .git-filen skal være med i oppgavemappen når dere leverer.

- ➔ Arbeidskravet må bestås for å ta eksamen i faget. Hele gruppen får “ikke bestått” dersom noe mangler, og må rette opp manglene sammen for å få godkjent.
- ➔ Alle hjelpemidler tillatt- men merk at **kopiering av kode fra AI eller andre kilder ikke er tillatt, det er plagiat.**
- ➔ Dersom et eller flere gruppemedlemmer av ulike årsaker ikke er delaktige i arbeidsprosessen, skal gruppen ta kontakt med Amina Brenneng og/eller Abebe Bediye.

Tekniske krav

- ➔ Dere skal benytte dere av følgende API: https://swapi.dev/
- ➔ Dere skal benytte Vanilla JS, HTML og CSS. Jest brukes til testing. Dersom dere bruker noen eksterne pakker utover dette skal det begrunnes i videobesvarelsen.
- ➔ Gruppen er ansvarlig for å sørge for at tester og programmet kjører slik det skal, samt at mappestruktur og filene vedlagt i besvarelsen er korrekte og ryddet opp i. En uorganisert besvarelse vil ha negativ påvirkning på sensuren.
- ➔ Sensor kommer til å kjøre: **“npm install” -> “npm test”** for å kjøre testene.
- ➔ Det er ikke et krav at nettsiden er responsiv for mobil og nettbrett, men den skal være egnet for PC.
- ➔ Det stilles ingen krav til styling utover det som er spesifisert i oppgavene.
- ➔ Gruppen har frihet til å bestemme selv hvilke fremgangsmåter dere ønsker å benytte dere av med tanke på å manipulere DOM. Eksempler på dette kan være getElementById vs querySelector, classList vs className osv. Gruppen må derimot komme til enighet om hvilken standard som skal brukes av alle medlemmer i besvarelsen. Dette gjelder også hvordan dere skal navngi klassenavn og variabler. Gruppen velger selv strukturen til nettsiden. Dere kan ta dere kreative friheter som for eksempel å benytte slette-ikoner i stedet for knapper etc, så lenge funksjonaliteten fungerer.

Oppgavetekst

Merk at begge sidene og deres underpunkter skal kodes.

Side 1 - Karakterer

Dere skal lage en oversikt over alle karakterer i Star Wars-universet. Det skal være mulig å filtrere alle karakterer etter Species, og man skal kun få opp de karakterene av den type Species man har valgt.

- ➔ **Hver karakter skal vises frem i hvert sitt kort, hvert kort skal vise følgende informasjon:**
  - ◆ Navn
  - ◆ Fødselsår
  - ◆ Hvilke filmer karakteren har vært med i
  - ◆ Species

- ➔ **Hvert kort skal inneholde en Slette-knapp og en Redigerings-knapp.**

- ➔ **Kortet skal ha en bakgrunnsfarge som representerer sin Species**. For eksempel, alle karakterer av typen “Wookie” kan ha en bakgrunnsfarge som er rød.

- ➔ **Lag et form på siden hvor brukeren skal kunne lage sin egen karakter.** Brukeren skal kunne fylle ut følgende felter:
  - ◆ Navn
  - ◆ Fødselsår
  - ◆ Species

- ➔ **CRUD-Operasjoner:**
  For hver CRUD-operasjon skal du skrive ut en console.log/console.error som viser HTTP-responskoden og beskjed for hvordan operasjonen gikk.
  - ◆ Create: Brukeren skal få lage en ny karakter ved å fylle ut formet. Denne karakteren skal vises frem på siden, lagres i CrudAPI og lagres i localStorage. Den nye karakteren skal listes opp sammen med de andre av sin Species.
  - ◆ Read: Alle karakterer skal vises frem på nettsiden, og kunne filtreres etter Species.
  - ◆ Update: Med redigerings-knappen skal man kunne redigere hvilken som helst karakter. Når man redigerer en karakter, skal karakteren oppdateres i localStorage og CrudApi. Brukeren skal kun kunne redigere Species til en annen Species som eksisterer i API’et. skal altså ikke være mulig å endre Species til “Enhjørning”, fordi dette er ikke en eksisterende Species. Hvis

brukeren endret Species på karakteren, må også bakgrunnsfargen til karakterens kort forandres samt at den må filtreres riktig.

- Man skal kunne redigere følgende felt:
  - Navn
  - Species

◆ Delete: Med slette-knappen skal man kunne slette hvilken som helst karakter. Dette skal oppdateres på nettsiden, i localStorage og CrudAPI.

➔ **Lag minst 5 tester i Jest**

Side 2 - Kjøretøy

Brukeren skal kunne bla gjennom forskjellige kjøretøy, og kjøpe de hen liker best
For hver CRUD-operasjon skal du skrive ut en console.log/console.error som viser HTTP-responskoden og beskjed for hvordan operasjonen gikk.

➔ **Nettsiden skal vise en oversikt over hvor mye brukeren har på konto.** Brukeren starter med 500.000,- credits, summen på konto skal lagres i localStorage og i CrudAPI.

➔ **Siden skal ha to seksjoner**: En seksjon som viser kjøretøy, og den andre seksjonen viser kjøretøyene brukeren eier.

➔ **I seksjonen for å vise kjøretøy skal det kun vises 6 kjøretøy om gangen på nettsiden i hvert sitt kort**. Ett kort skal inneholde:

◆ Name
◆ Model
◆ Cargo Capacity
◆ Cost in credits

➔ **Hvert kort skal inneholde en "Kjøp"-knapp.**

◆ Om brukeren kjøper et kjøretøy, skal kontoen bli trukket med hva kjøretøyet koster i credits. Lommeboken må oppdateres i localStorage og CrudAPI. Kjøretøyet skal da legges i seksjonen som viser hvilke kjøretøy brukeren eier.
◆ Kjøretøy som kjøpes skal postes opp i CrudAPI og lagres i localStorage. Når man refresher siden, skal man fortsatt se kjøretøyene man eier.
◆ Det er ikke mulig å kjøpe samme kjøretøy 2 ganger.

➔ **Brukeren kan trykke på piltasten -> for å få frem de 6 neste kjøretøyene.** Dersom hen trykker på piltasten tilbake <- vises de 6 forrige kjøretøyene.

➔ **Brukeren kan ikke kjøpe flere kjøretøy enn det hen har råd til.**

- ➔ **I oversikten over kjøpte kjøretøy skal man kunne selge hvert kjøretøy man har kjøpt.**
  - ◆ Hvert kjøretøy man eier har en “Selg”-knapp.
  - ◆ Når man selger et kjøretøy, får man bare solgt det til 80% av innkjøpsprisen.
  - ◆ Lommeboken og kjøretøy skal oppdateres både på nettsiden, localStorage og CrudAPI.
  - ◆ Det skal være mulig å kjøpe kjøretøyet igjen når det har blitt solgt.
  - ◆ Kjøretøyet skal forsvinne fra oversikten over kjøpte kjøretøy.
- ➔ **Easter egg!** Legg inn en hemmelig funksjonalitet for at brukeren skal kunne tjene penger. 🤑
- ➔ **Lag minst 5 tester i Jest**

Lykke til! 🎉