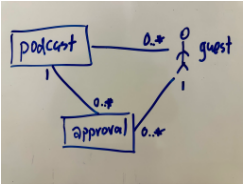


			Løsningsforslag Modul B eksamen vår 2024 (2024-06-26, v. 1.0) Vedlagt følger momenter ved skåring og løsning av oppgavene i årets eksamen. Jeg har valgt å skrive ut tankeprosessen rundt hver oppgave i større grad i år for å gi bedre innblikk i tanker jeg håper studenter kan lære noe av. Jeg har således fokusert på kvantitet i tilbakemeldingen framfor å skrive kort og med få skrivefeil. Er det noe vesentlig som mangler, si ifra! Gunnar (gunnab@uio.no)		
#	poenng	oppgave navn	oppgave tekst	sensurveiledning	løsningsforslag
4a	3	"Hylleware"	Gjør kort og punktvis rede for andre problemstillinger utover lagring (nevnt over) som du ser ved å velge "hylleware" i prosjektet. Benytt formen [problem:] [kort beskrivelse av problem] i ditt svar. Gjør deretter kort og punktvis rede for vanlige fordeler ved å velge "hylleware" generelt. (3 poeng)	Oppgaven ber eksplisitt om punktvis og kort redegjørelse for (i) problemstillinger i caset spesifikt og deretter (ii) fordeler generelt ved å velge hylleware. Sentralt for begge punkter er hvorvidt man kan anvende jerntriangelet (tid, kostnad, funksjonalitet) og vurdere dette opp mot fordeler og ulemper ved å kjøpe en ferdig løsning. Det er ønskelig med en svarform som kunne gått inn i et lysark i en prosjektpresentasjon hvor agenda er hvorvidt man skal bruke hylleware eller ikke. For (i) kan nevnes problemer rundt begrenset funksjonalitet, vanskelig integrasjon (tid), uoversiktlig kostnad over tid, samt andre elementer man muligens har fått med seg i modulen: juridiske problemstillinger eller krav (GDPR, WCAG, med mer), redundans av funksjonalitet, sikkerhet, oversettelsesproblemer til norsk, og at man vil være låst til en leverandør. For (ii) fordeler kan nevnes de samme momentene som i (i) med motsatt fortegn, muligens med ekstra fokus på at det kan være veldig raskt å komme i gang (tid) med en kjent kostnad hvis tilstrekkelig funksjonalitet er tilstede i hyllewaren. I tillegg vil et ferdig system ofte vil ha færre bugs og man kan få support fra erfarne utviklere. Et ferdig system vil også typisk ha funksjonalitet man kanskje ikke har tenkt igjennom at man vil komme til å trenge framover. Andre momenter som er fornuftige kan også være poenggivende. -- For full skår er det forventet minst tre fornuftige punkter for (i) og tre fornuftige punkter for (ii).	[Se sensurveiledning for punkter. Som oppgaveteksten beskriver, det er ønskelig med punktvis oppstilling, f.eks.:] <i>Ulemper:</i> <i>*Begrenset funksjonalitet: Usikkert hvorvidt hyllewareløsningen kan levere funksjonalitet som løser de viktigste usecaseene i prosjektet.</i> <i>*Begrenset integrasjon: Det kan være vanskelig å få hyllewaren til å fungere med andre eksisterende systemer i Oslo skolen.</i> [...]
4b	2	Åpen kildekode	Ta utgangspunkt i de perspektivene som har vært dekket av forelesninger og pensum i kurset, og gjør kort og punktvis rede for fordeler og ulemper ved at LP-prosjektet bruker open-source kodebasen fra det overnevnte GitHub-prosjektet når første versjon av systemet skal utvikles. (2 poeng)	Oppgaven omhandler bruk av eksisterende åpen kildekode (Sommerville 7.4) i lys av de tre elementene fra jerntriangelet (se forrige oppgave) i punktvis form (fordeler/ulemp(er)). Det er ønskelig med en svarform som kunne gått rett inn i et lysark i en prosjektpresentasjon hvor agenda for møtet er hvorvidt man skal bruke åpen kildekode eller ikke. For fordeler (i) kan nevnes lavere kostnad, "time to market" (tid), "new features over time" (funksjonalitet) i tillegg til bedre "community support", transparens, sikkerhet (bedre testet, men også mer åpent som kan være en ulempe) og tilgang til utviklere som kjenner teknologien. For ulemper (ii) kan nevnes avhengigheter typiske elementer knyttet til funksjonalitet, variabel kvalitet, vanskelig å integrere, vanskelig oppsett mot tredjepartssystemer og sikkerhet. Det kan også være juridiske og copyright utfordringer samt hvorvidt det åpne kildekodeprosjektet i framtiden vil fortsette å være aktivt eller sakte dø. -- For full skår er det forventet minst to fornuftige punkter for (i) og to fornuftige punkter for (ii).	[Se sensurveiledning for punkter og forrige oppgave for ønskelig punktvis oppsett.]
4c	2	Krav	[...] Angi hvilke krav over som er funksjonelle og hvilke som er ikke-funksjonelle, for eksempel, slik: "Funksjonelle: [liste med nummer] og ikke-funksjonelle: [liste med nummer]".	Oppgaven ber kandidaten gjøre konkrete valg (dvs. diskutere) om hvilken type krav som presenteres i caset. En tenkt realistisk jobbsituasjon er at kandidaten blir bedt om å strukturere og gruppere krav som er avdekket i et prosjekt under kravinnhentingprosessen (evt. i løpet av en sprint). -- Oppgaven skåres ved å telle feil kategori for hvert krav: Ikke-funksjonelle krav er 1, 6, 9, 12, og 13. Resten er funksjonelle. Null eller én feil gir full uttelling og to til fire feil gir halv uttelling. (Merk at man ved å svare tilfeldig vil man ha forventet 0.5 feil hvis man antar at de 13 kravene må tilhøre en av de to nevnte typene).	~~Funksjonelle:~~ 1, 6, 9, 12, 13 ~~Ikke-funksjonelle:~~ 2, 3, 4, 5, 7, 8, 10, 11
4d	1	Systemkrav	Er det noen av kravene over som er systemkrav? Angi i så fall nummer.	Oppgaven har samme kontekst som 4c og vurder om kandidaten har et bevisst forhold til hva som er krav til selve systemet. -- Oppgaven skåres ved å telle feil kategori for hvert krav: Systemkrav er krav 1, 6, og 12. Maksimalt to feil gir full uttelling, ellers ingen uttelling.	<i>Systemkrav:</i> 1, 6, 12
4e	2	Prioriteringer / priorities	Det er et vesentlig poeng at første versjonen av systemet i LP-prosjektet kan prøves ut av elever på noen få skoler så raskt som mulig. Hvilke av kravene over mener du bør utelates i første versjon av løsningen? Angi nummer. Hvis du er i tvil om et krav kan utelates, bruk én setning på å forklare hvorfor.	Oppgaven ser om kandidaten kan prioritere viktigheten av forskjellige krav i et case. Prioriteringer er helt sentralt innen all programvareutvikling og listen av store prosjekter i industrien som har feilet fordi man ikke klarer å prioritere krav er lang. Prioriteringer er også viktig å kommunisere til kunde, f.eks. "Jeg skjønner du ønsker jeg skal gjøre 100 ting, men hvilke av disse tingene er det viktigst for deg at jeg starter med?" Oppgaven skåres ved å telle feil kategori for hvert krav på formen 2 minus hver feil: Krav som kan utelates i første versjon: 4, 13. Krav 5 sees bort fra da det kom inn gode innspill fra kandidater under eksamen på at kravet både kan droppes (selv om det opprinnelig var planlagt som et hint til hvordan domenemodellen kan se ut.) Uansett, takk til dere som påpekte dette! :) Forøvrig kan uttrykt tvil (dvs. en setning) om et krav vurderes som 0.5 feil ved stryktilfeller. Dette da kandidaten ser problemet med om kravet kan (eller ikke kan) droppes.	<i>Kan utelates i første versjon: 4, 13. (Krav 5 er jeg i tvil om.)</i> •

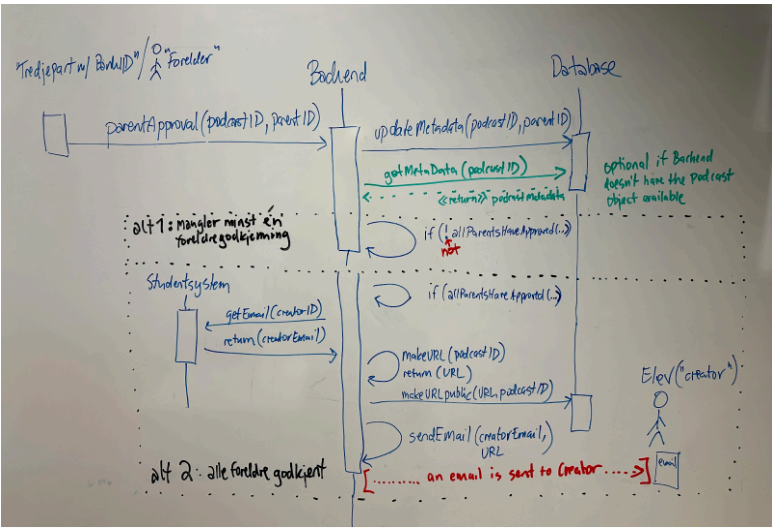
4f	8	Domenemodell	[...] Gjør kort rede for problemer med GPT-4 sitt forslag til løsnings punktvis. Benytt en synkende grad av viktighet. Kommenter også kort (3-4 setninger) hva du ville gjort annerledes enn det som foreslås av GPT-4.	Oppgaven tar utgangspunkt i en av de viktigste modellene i kurset, nemlig domenemodellen. Domenet som presenteres i caset er podkasts i kontekst av de utfordringer som eksisterer rundt foreldresamtak for elever under 16 år. Viktige use-cases i et prosjekt vil nesten alltid sjekkes opp mot om man kan finne en "løsning" på use-caset teknisk ved bruk av domenemodellen. Kan man ikke det, har man minst ett problem som garantert kommer til å bli mer alvorlig etter hvert da man må enten droppe krav som er viktig eller endre domenemodellen og deretter fikse opp i alle kode som er skrevet allerede som basert på en feil i domenemodell. Oppgaven tar i tillegg inn bruk av resultater generert i en stor språkmodell som garantert kommer til å ha relevanse for IT-utviklere i mange år framover. GPTs forslag til domenemodell inneholder flere større og vesentlige innholdsmessige (semantiske) feil i tillegg til mindre (syntaktiske) feil og andre misforståelser. Noen av feilene kommer av mangel på presisjon i "prompt", men all erfaring viser at ingen skriver feilfrie prompt de første gangene når det som skal løses er ikke-trivielt. En god besvarelse vil påpeke vesentlige svakheter ved GPT i tilstrekkelig grad til at det burde være klart at det hadde vært bedre å lage en domenemodell selv. GPT klarer riktignok å få ut klasser og relasjoner som tekst med utgangspunkt i promptet, men diagrammet er ganske håpløst. Av store og vesentlig feil i GPT forslag er at modellen har tre grupperinger av klasser som ikke er sammenkoblet via multiplisitet og at Approval og ParentApproval uttrykker det samme (er duplisert) og må åpenbart inngå i den samme klassen. Basert på disse to store feil inneholder modellen følgelig også en drøss med mindre feil (ikke uttømmende) som feil multiplisitet, manglende multiplisitet, unødig kompleksitet rundt modellering av arv for student/creator/guest, URL kravet er ikke i løsning, med mer. Av delspørsmålet på hva som burde vært gjort annerledes kan det nevnes at det eksisterer allerede et Studentsystem hvor både elever og foreldre kan antas å ligge og at det ville være lite hensiktsmessig og totalt unødvendig for LP prosjektet å bygge et nytt studentadministrasjonssystem. LP prosjektet bør isteden kun fokusere på å levere funksjonalitet som skolens eksisterende systemer ikke kan levere i konteksten av distribusjon og foreledregodkjenning av podcasts. Videre bør man ta utgangspunkt i å lage en domenemodell av det sentrale i oppgaven på så enkel mulig måte som fortsatt tilfredsstiller kravene. Noen momenter følger her for å vise resonering rundt hva er "enklest mulig" for caset som er beskrevet. En enkel tilnærming vil da være å i alle fall innføre klassen Podcast (caset handler om podkast) og deretter spørre seg "hva er det man absolutt må ha i tillegg til denne klassen?" Tja, (Parent)Approval er i henhold til kravene noe som er helt sentralt på grunn av lovverket som eksister. Hvordan forholder Parent approval seg til Podcast og student/Creator/Guest? Vel, vi vet at det er kun studenter som er gjester og at hver gjest således må ha minst én foreldre. Hver Approval må således være unikt koblet til én Podcast (multiplisitet = 1) og én student/Guest (multiplisitet = 1). Hver Student/Guest kan delta på 0..* Podcast og må da ha medfølgende Approval for hver av disse, dvs: Podcast 1-----0..* Approval 0..*-----1 student/Guest. Dette forholdet kan løses feks. i Python ved bruk av et dictionary hvor et Student-Parent par utgjør at man har mottatt en "godkjenning", se bilde. (Det kan argumenteres at man ville utvidet dette til å også inneholde metadata fra godkjenningen, men dette er ikke nevnt som krav og kan droppes i første versjon.) Hvis man i tillegg benytter en liste for Guests, burde det være rett fram å skrive et metode i Podcast som sjekker om alle Guests har Approval. Merk forøvrig at problemstillinger hva personID, studentID faktisk er er ikke det sentrale for oppgaven slik at momenter rundt dette trekker verken opp eller ned. Uten at jeg kjenner detaljene vil jeg hevde at Osloskolen har garantert en lovlig og hensiktsmessig måte for å unikt representere hver student og forelder i sine systemer. LP prosjektet burde ikke prøve å endre på noe av dette, snarere benytte akkurat den samme tilnærmingen som Osloskolen benytter fra før. Gitt at vi ikke vet hva den riktige representasjonen er, la oss derfor bare kalle dette for en "ID" som vi vet at unikt identifiserer hver person. — Oppgaven skåres med utgangspunkt i at man får fire poeng ved å uttrykke skepsis (fortrinnsvis sterk skepsis) til GPTs forslag samt at man begrunner dette ut ifra minst ett vesentlig moment (nevnt over). Deretter justeres poengsum for mengden av meningsfylte momenter. Av mindre momenter er at bl.a. at url mangler, trøbbel med multiplisitet for perifere klasser og arv. For arv mener jeg ting kunne vært løst uten arv først for å gjøre ting enkelt og deretter legge til arv hvis domenemodellen åpenbart trenger dette for å løse nye usecases.	En domenemodell viser hvilke klasser som eksisterer i et system og deres relasjoner. GPTs forslag inneholder flere vesentlige feil sånn som at * foreldregodkjenning er duplisert (ParentApproval og Approval er det samme) og at * diagrammet videre har tre grupperinger av klasser som ikke er sammenkoblet (hva er i så fall deres relasjoner)? Disse problemene i tillegg til GPTs * manglende forståelse av hva den underliggende modellen er medfører blant annet at * den tegner noe som ser ut som en domenemodell, men piler og multiplisitet virker å være kastet inn litt tilfeldig. * Modellen forelår også arv, som jeg ikke mener er nødvendig i caset. * Av mindre momenter fikk den ikke med seg URL kravet som jeg tror bare var forvirrende for GPT. Det jeg ville gjort annerledes er å ta utgangspunkt i min egen forståelse av caset hvor jeg mener det er to vesentlige klasser som må inn i en domenemodell: Podcast og Approval. Jeg kan ikke se hvordan caset løses uten disse klassene. Osloskolen har allerede et Studentsystem hvor hvor jeg antar både studenter og deres foreldre er representert og det eneste jeg trenger i domenemodellen er unike identifikatorer på hver student (studentID og parentID) fra dette systemet og putte dette inn i Podcast og Approval. Multiplisitet for de to sentrale klassene er at hver Podcast må ha én og bare en Creator og må ha 0..* Approvals. Hver Approval må være koblet til én student. På denne måten vil hver Guest (dvs. en Student under 16 år) kunne ha 0..* Approvals.
----	---	--------------	--	--	---



```
class Podcast:
    def __init__(self, creator):
        self._creator = creator
        self._guests = [] # List of student_id
        self._approvals = {} # Dict of student_id (key) and parent_id (value) that gives approval
```

4g	10	Tekstlig use-case	Tekstlige usecases er en fin måte å nærme seg hvordan en user story (som en <<type bruker>> ønsker jeg <<et mål, en funksjon >> slik at jeg kan <<en begrunnelse>>) kan brytes ned i steg som er tettere knyttet mot selve implementasjonen i systemet. Ved at man også må beskrive (i) aktører (primære og sekundære), (ii) prebetingelser (hva antas før caset begynner) og (iii) postbetingelser (hva er resultatet av at caset "kjøres") får man også tydeliggjort input og output av prosessen. Denne øvelsen kan også være en fin måte å begynne å tenke rundt hvilke ansvarsområder forskjellige deler av systemet har, samt hva systemet vet om og ikke vet om omliggende systemer, klasser med mer. Oppgaven spør feilaktig etter "en (sic) tekstlige beskrivelser ..." av to use cases og blander således entall og flertall (merk at oppgavenavn og også engelsk oversettelse benytter flertall). Poenggivning skiller derfor ikke på om de to use-casene beskrives samlet eller separat (to use-cases). Formkravet til en tekstlig use-case er det samme, uavhengig av om casene ble beskrevet samlet eller hver for seg: aktører, prebetingelser, hovedflyt, evt. alternativ flyt, og postbetingelser. Merk forøvrig at de to casene er mest hensiktsmessig å behandle separat da primæraktør (den som er den aktive i caset) i hvert case er forskjellig: Når en forelder initierer en godkjenning vil denne forøvrig komme inn i LP systemet via sekundæraktør (med bankID) og i tillegg kreve oppslag av e-post hos sekundæraktøren Studentsystem. Det er kun forventet at hovedflyt inneholder noen grovt sett riktige elementer og det trekkes kun hvis feil er vesentlige avvik fra overordnet prosess fra en podcast blir registrert av Creator til en epost med url er mottatt. — Oppgaven skåres ved å først vurdere om formkrav til et tekstlig use case er tilstede. Poeng som følger hvis et usecase benyttes (del poeng på to hvis to use case er beskrevet). Først gi 4 poeng hvis navn, pre, post, hoved og alternativ flyt er tilstede og knyttet til case da dette viser at man er kjent med formkrav. Deretter gis opp til 4 ekstra poeng for at man faktisk har brukbart korrekte steg som trengs for å oppnå postbetingelsene. Ekstrapoeng for alternativ flyt i det andre caset samt flere poeng ting faktisk hadde fungert eller for refleksjoner rundt momenter som vurderes tatt med på en god måte. Merk at jeg har tatt med hvem av Frontend og Backend som gjør hva, samt hva tredjepartssystemer gjør i løsningsforslaget. Dette er ikke påkrevt, men kan være lurt å forholde seg til som forbedredelse til senere sekvensdiagram hvor alle systemer (objekter) må angis eksplisitt (i tillegg til andre momenter som også må inn).	[Se også sensurveiledning for dette spørsmålet samt stegvis oversikt i sensurveiledning for neste spørsmål 4h] <i>Usecase: "Elev legger inn podcast for godkjenning"</i> <i>Aktører: elev (primær), Studentsystem (sekundæraktør for oppslag av ID for gjester), "foreldregodkjenningssystem" (som bruker BankID); Parent agerer forøvrig ikke direkte på LP.</i> <i>Pre: elev er logget inn, har en ferdigprodusert podcast som er lastet opp og har lagt inn gjester som alle er riktig identifisert i Studentsystem</i> <i>Post: alle foreldre til gjester i podcast mottar forespørsel om godkjenning og podkast er lagret i database</i> <i>Hovedflyt:</i> <i>1. Elev trykker på "Registrer podcast for godkjenning" i Frontend.</i> <i>2. Backend gjør oppslag i Studentsystem på hvem foreldrene til Guests er</i> <i>3. Backend oppretter en unik PodcastID, setter Creator, GuestID og lagrer podcast info og metadata i Database</i> <i>4. Backend sender én forespørsel til tredjepartssystem med parametere PodcastID og ParentID for hver Guest (antar dette er tilstrekkelig da det er denne informasjonen tredjepartssystemet sender tilbake i neste usecase)</i> <i>Alternativ flyt: ingen (ser bort ifra feilhåndtering da jeg ikke oppfatter dette som sentral siden det ikke er nevnt med et ord i teksten)</i> <i>—</i> <i>Usecase: "Forelder godkjenner en podcast"</i> <i>Aktører: "foreldregodkjenningssystem/BankID" (evt. også forelder) som primæraktør, Studentsystem (sekundær) og elev (agerer ikke på systemet, mottar kun en epost)</i> <i>Pre: En eller flere foreldre har akkurat godkjent en podkast tredjepartssystem; podkast er registrert i systemet</i> <i>Post: elev har mottatt url til podcast og podcast ligger tilgjengelig på internett (det er fordel at alle godkjenninger av foreldre er lagret i database som metadata, noe som er nødvendig for å vite om alle foreldre har godkjent nå en ny godkjenning foreligger)</i> <i>Hovedflyt:</i> <i>1. Backend mottar kall i sitt API om at en forelder har akkurat godkjent (Approval)</i> <i>[Backend henter metadata for riktig Podcast i Database hvis ikke denne allerede er tilgjengelig]</i> <i>2. Backend lagrer Approval i Database</i> <i>3. Backend sjekker om alle Guests nå har Approval</i> <i>Alt1: Det er forsatt foreldre som ikke har godkjent</i> <i>4: Gjør ingenting, vent videre på nye Approvals</i> <i>Alt2: Alle foreldre har nå godkjent</i> <i>4. Backend spør Studentsystem om epost til Creator</i> <i>5. Backend genererer url for Podcast</i> <i>6. Backend lagrer url som metadata i Database</i> <i>7. Backend sender epost med url til Creator</i>
----	----	-------------------	---	---

4h	8	Sekvensdiagram	Modellér use-caset "forelder godkjenner en podkast" (det samme use-caset du beskrev tekstlig i forrige oppgave) som et sekvensdiagram. Merk at riktig oppførsel for systemet er at den som lagde podkasten (Creator) skal kun motta en e-post med link til podcast hvis alle gjester (elever) sine respektive foreldre har godkjent deling. I henhold til beskrivelsen av use-caset vil LP-systemet som skal utvikles muligens kreve flere nye metoder i noen komponenter i systemet. Disse metodene skal føres inn i sekvensdiagrammet. Under er en oversikt over metodekall eller API kall mot tredjepartssystemer som du kan benytte deg av i sekvensdiagrammet: Podkastsystemets backend har et grensesnitt (API) som et tredjepartssystem kan benytte når en forelder har logget inn med BankID og godkjent en spesifikk podkast, som skjer ved at tredjepartssystemet kaller parentApproval(podcastID, parentID) i backend. For å finne e-post til en student kan du bruke tredjepartssystemet "Studentsystem" sitt API via getEmail(studentID) som returnerer en e-postadresse. Podkastsystemets database har metoden makeURLpublic(URL, podcastID) som tar en URL og en podcastID og lagrer dette som metadata i databasen samt sørger for at podkasten kan streames på internett. For å spare tid anbefales det at du tegner sekvensdiagrammet på et skisseark, men du kan legge til eventuelle kommentarer under.	Oppgaven omhandler sekvensdiagram for et usecase som kandidaten allerede har beskrevet tekstlig i tidligere oppgave. Oppgavene har således et overlapp da de beskriver "det samme" men på forskjellige måter (et aktivitetsdiagram vil også kunne beskrive noe av det samme som sekvensdiagrammet). Caset beskriver i tekstlig form vordan det er forventet at LP systemet skal oppføre seg samt hvilke andre systemer som eksister som systemet kan kommunisere med. Hvorvidt en kandidat klarer omsette denne informasjonen om caset til en bedre strukturert og en mer presis form avdekkes i denne oppgaven. Det er forventet at kandidaten skjønner hvem som agerer i use-caset. Den fysiske handlingen med å godkjenne vil være fra en forelder, men handlingen til forelder eksisterer ikke i LP systemet. Det vil derimot være et tredjepartssystem som kaller Backend sitt API som er den som agerer. Tredjepartssystemet er således den primæraktøren i caset. Dette tilnærmingen ble valgt i årets eksamen da jeg så på eksamen i fjor at altfor mange studenter tror at alle sekvenser starter at en bruker logger inn. Dette er feil. Login vil være et viktig usecase for systemer som har brukere, men det betyr ikke at login må repeteres hver gang et case modelleres (dette er grunnen til at man har prebetingelser i tekstlige use-cases). Selve det som skal gjøres er ikke komplekst, men utfordringen med å skille ut hva som er vesentlig for sekvensdiagrammet i en lengre tekstbolk kan være krevende, spesielt hvis man ikke vet hva et sekvensdiagram er og hva det mer ment å formidle til leser. — Oppgaven skåres som 0 ved ubesvart eller hvis det er ingenting som kan minne om aktører (horisontalt) eller sekvenser (vertikalt) i diagrammet. 1 poeng gis hvis det virker sannsynlig at man skjønner at "går" bortover og noe går nedover (dvs. man ville kanskje gjenkjent et sekvensdiagram blant andre diagramtyper). 4 poeng gis hvis hovedformkrav til et sekvensdiagram er tilstede selv om løsningen ikke gir særlig mening. +2 poeng gis for at deler av sekvensene gir mening, altså noe av løsningen kunne vært brukt. Feil diagramtype men ellers stegvis løsning av case vil typisk ligge under 4 poeng. Alternativ flyt gir ekstrapoeng.
5a	3	Organisering	Diskuter kort hovedforskjellene mellom en tidsboksbasert prosessmodell (Scrum) og en flytbasert prosessmodell (Kanban).	Oppgaven tar utgangspunkt i om man vet noe mer om scrum og kanban utover at den ene er tidsboksbasert og den andre er flytbasert. Oppgaven tar ikke utgangspunkt i caset, men momenter i dette spørsmålet vil flyte inn i det konkrete caset i 5b. — Oppgaven skåres med 0.5 poeng for hvert moment for hver model, maks 1.5 per modell. Flyt vs tidsboks framgår av oppgaveteksten og gir ikke poeng, men følgende momenter er poenggivende (ikke uttømmende liste): definerte roller (scrum), visuelle states (kanban, DOING, etc.), rigid(scrum) vs fleksibilitet (kanban), bokser/deadlines/leveringer (scrum, tid låst) vs throughput (kanban, functionality), WIP (kanban), seremonier (scrum), regulert prosess (scrum), flaskehalser (kanban) Merk: ikke dobbelt poeng for samme moment, f.eks. "fleksibilitet" (scrum, +0.5) gir ikke også poeng for "rigid" (agile) og trekk for stikkordbasert opplisting (-0.5 til -1 da det spørres om "kort" ikke "stikkord")



[se sensurveiledning for punkter]

5b	3	Prosessmodell	Anbudssystemet for offentlige anskaffelser i Norge krever at alle kontrakter over en viss totalsum må legges ut på anbud. Anbud er en krevende og kostbar prosess som tar mye tid. Alt arbeidet som så langt er utført i LP-prosjektet er utført av fast ansatte med gode tekniske ferdigheter. Men andre personer (ekstern bistand) må inn i LP-prosjektet for å utvikle den siste delen av løsningen da ingen er komfortable med å utvikle distribusjonssystemet for podcaster blant de ansatte. Velg én generell prosessmodell for programvareutvikling som du mener bør benyttes i prosjektet, gjerne ut fra interessentenes behov, ønsker, eller preferanser som framgår i oppgaven. Begrunn også hvorfor du ikke valgte andre modeller.	Opgaven bygger omhandler om man kan anvende kunnskap om forskjeller i prosessmodeller i caset. Det er lagt opp til at fossefall er svaret da det er liten usikkerhet rundt hvordan første versjon av systemet skal se ut og det er ønskelig å benytte en leverandør som kan levere den nødvendige og begrensede funksjonaliteten innen den veldig korte tilgjengelige tiden. Merk at oppgaven eksplisitt pressiserer at eksisterende team har god kontroll på prosjektet. Oppgaven tar inn en drøfting av andre modeller (scrum, kanban) for å samlet vurdere alle hovedalternativer opp mot hverandre og på den måten få vist at man ser fordeler og ulemper ved hver prosessmodell, fortrinnsvis knyttet opp mot behov i caset. – Opgaven skåres med utgangspunkt i 0 poeng, deretter poeng hvis alle tre modellene er nevnt (+1) og drøftet for caset (+1) samt at fossefall er svaret (+1) eller i det meste veldig fristende.	<i>Jeg velger fossefall da både prosjektet og kravene er veldefinerte og en god del av arbeidet er ferdig slik at resten av jobben kan avtales med en leverandør at må leveres før tidsfristen.</i> <i>Tidspresset som er involvert i prosjektet medfører at å legge hvert krav i en kø (Kanban) og håpe at man i sum blir ferdig før tiden er ikke en god tilnærming. Her virker det bedre å ha tett styring på avhengigheter mellom leveringer slik at leverandøren kan sette flere folk på å få alt ferdig innen fristen</i> <i>For senere versjoner er det imidlertid åpenbart at prosessmodell kan revurderes avhengig av om det er klart hva som skal bygges eller ikke. Her kan Scrum fungere godt hvis det er uklart hvilke usecases som skaper mest nytte for produkteier og man samtidig ønsker en tettere kobling mellom utviklerteam og Osloskolen.</i>
5c	2	Kontraktype	Siden det meste av kravspesifikasjonen allerede er utført kan det friste med en fastpriskontrakt for å ha kontroll på utgiftene og for å sikre at systemet leveres innen skolestart om to måneder. Bøter til leverandøren kan da benyttes ved for sen levering. Et alternativ er å leie inn bistand (konsulenter) for den samme kontraktsummen. List opp argumenter for og imot to forskjellige typer kontrakter som kan brukes i LP prosjektet: fastpriskontrakt og kontrakt om bistandskjøp.	Opgaven tar utgangspunkt i oppgave 5b og bygger videre på hvordan oppdraget skal reguleres kontraktsmessig med en leverandør. To alternativer blir presentert (fastpris og bistand) og oppgaven spør om argumenter for og imot begge. Caset så langt skisserer låst pris (pga anbud) og omfang (låste krav), men det er også andre momenter som bør inn. Som vanlig er jerntriangelet "vår venn" og bør gi kandidaten momenter å diskutere for hvert alternativ. – Opgaven skåres som to poeng ved minst en fordel og minst en ulempe for hvert av alternativene; eksempler på gode momenter følger under: fastpris fordel: pris, scope, oppfølging av leverandør, insentiv til å levere rask, forutsigbarhet, leverandør har hovedansvar fastpris ulempe: mangel på fleksibilitet, kutte hjørner/produktkvalitet, mye planlegging up-front (egentlig ikke et problem her), krangling om kontrakt ressurskjøp fordel: fleksibelt (egentlig ikke nødvendig), konsulent får tettere samarbeid med eksisterende teknisk team, ressurskjøp ulempe: ikke kontroll på budsjett, ansvar for onboarding og oppfølging av konsulent, mer tid brukt på oppfølging	[Se sensurveiledning for punkter]
6b	1	Sikkerhet	Kort tid etter at den første versjonen av systemet som ble bygget i LP prosjektet ligger ute på en lokal skole viser det seg at systemet ved flere anledninger slutter å respondere. En gjennomgang av systemlogger viser seg at noen har laget et skript som laster opp en podkast hundrevis av ganger i sekundet i frontend rett før hver gang systemet slutter å respondere. Loggene viser også at en fiktiv studentID, som brukes til å teste systemet, er blitt benyttet ved innlogging i produksjonsmiljøet til LP. Med ett ord, beskriv hvilket sikkerhetsmål denne aktiviteten er et brudd på.	Opgaven spør etter hovedproblemet ("ett ord") for sikkerhetsbruddet, som er knyttet til at systemet slutter å være tilgjengelig (T) for studentene. Tilgjengelighet er er en av de tre hovedtypene brudd, de to andre er Konfidensialitet og Integritet (KIT, evt. CIA på engelsk). Full skår for Tilgjengelighet. Integritet kan sees som et halvt poeng for kandidater som ligger rundt strykgrensen.	<i>Tilgjengelighet</i>