Eksamen har til hensikt å teste studenten i følgende områder:

1. Design:

- Lage UML diagram for å vise hvordan man tolker datamodellen.
- Gjøre nødvendige endringer i design basert på endring i scope (eksamen utvikler seg i omfang)

2. Frontend:

- Implementere React frontend i ulike steg basert på hva oppgaven krever
    - Vise hensiktsmessige oppdeling av komponenter og bruk av props
    - Vise at man forstår hvordan koble seg på livssyklusen til en React applikasjon
    - Håndtere state både lokalt og globalt
    - Vise at man forstår hvordan parent/child forholdet virker i React
    - Vise hvordan man kan lytte til events og endre UI basert på events (eks. onClick)
    - Vise at man kjenner til ulike *hooks* og hvordan de kan brukes
- Bruke axios/fetch til å hente data fra API
    - Dele opp i hensiktsmessige servicer
    - Håndtere eventuelle feil som kan oppstå når man henter data via API
- Benytte "PropTypes" for å validere props (ikke vist i presentasjon så trekker kun opp)
- Vise at man behersker ulike former for testing med Jest og enzym/react-testing-library. Det innebærer testing av et utvalg:
    - Komponenter med og uten props
    - Komponenter som henter data fra API
    - Hjelpefunksjoner som benyttes
    - Vise hvordan sette opp test-coverage
- Vise hvordan man bruker Styled Components til å implementere et design
    - Dynamisk kunne oppdatere komponenter sin stil basert på props
    - Gjenbruk av "base" stiler for å lage nye

3. Backend

- Implementere Node med Express i ulike steg basert på hva oppgaven krever
    - Vise hensiktsmessig oppdeling av applikasjonen
    - Vise at man forstår Route ⇒ Controller ⇒ Service ⇒ Modell
    - Vise hvordan man bruker eksterne biblioteker til å løse ulike oppgaver

- Bruke mellomvare til å håndtere autentisering/autorisering og validering av data
    - Vise at man forstår hva global mellomvare er
    - Vise hvordan lage og bruke custom mellomvare
- Forstå hvordan håndtere feil som kan oppstå i applikasjonen, globalt og lokalt
- Bruke Jest til å teste endepunkter med eller uten testdatabase.
- Vise at man forstå hva CRUD er og hvordan implementere dette med hensiktsmessige ressurser, metoder/verb og responskoder/type
- Vise hvordan man kan jobbe med epost i Node
- Vise hvordan man kan jobbe med filopplasting i Node
- Vise hvordan man kan eksportere data som CSV/EXCEL
- Vise hvordan man kan forhindre grunnleggende sikkerhetsutfordringer knyttet til
    - Hashing av passord
    - CSRF
    - XSS
    - NoSql Injection
    - Sanering av data
    - Environmentvariabler
- Vise at man kan søke i, filtrere og paginere data
- Dokumentere API i Postman

4. Database

- Sette opp MongoDb lokalt eller i skyen med hensiktsmessige schemas / modeller
- Vise at man kan lage relasjoner mellom modeller 1:N
- Vise at man kan CRUD data via ORM
- Vise at man forstår hva "hooks" betyr i Mongo sammenheng
- Vise at man kan aggregere data

5. Autentisering og autorisering

- Vise at man forstår kjente risikofaktorer knyttet til autentisering og autorisering
- Vise at man kan implementere frontend hvor brukeren kan lage bruker, logge inn og logge ut.
- Vise at man kan lage et grensesnitt som tar hensyn til om man er logget inn og hvilke rettigheter man har klientside og serverside
- Vise at man kan implementere backend som tilfredsstiller kravet til frontenden
    - Vise at man forstå hvordan kommunisere brukerrettigheter og tilgang mellom server og klient (session, cookie og / eller JWT)

6. Prosjektdokument

- Vise at man har fordelt oppgavene på en hensiktsmessige måte
- Vedlegg og link som viser at man har hatt god kommunikasjon/prosjektstyring via f.eks Trello/Azure Devops/Jira og GitHub/Bitbucket
- Ved behov forklare antagelser man har gjort i oppgaven sin
- Ved behov forklare utfordringer som oppstod og hvordan de ble løst
- Vedlegg som viser at man har forstått hvordan bruke pull requests, git commits og branching
- Vedlegg som viser hvordan man har delt opp prosjektet i epics og user stories
- Vedlegg som viser UML designet og kort forklaring til modellene og relasjonene
- Maks 1 A4 side + vedlegg

Legger ved en enkel mal for hva dokumentet kan inneholde.

7. Generelt

- God struktur teller positivt (mappestruktur, navngivning, dry m.m)
- Bruker du noen andre sine kode så skal den kommenteres og legges ved i readme (kommentere over koden du har brukt og pek til denne linjen i readme.md så det er lett å finne frem til)
- Vise at man klarer å levere en oppgave som er lett for mottakerne å forstå / sette seg inn i / starte
    - Bruke Readme.md til å forklare hvordan sette opp og starte applikasjonen

Karakterkrav

Kravene nedenfor er ikke absolutte men gir en pekepinn på hva som forventes. Helhetsinntrykket av oppgaven må også vektlegges. Kandidater kan med fordel vise økt forståelse/funksjonalitet for frontend som veier opp for manglende funksjonalitet/forståelse i backend.

Bør være på plass for å få minst E

- Prosjektdokument med tilhørende vedlegg
- Bruk av GIT
- En front-end i React som implementerer alle statiske sider
- Stilsett nettsiden ved hjelp av Styled Components.
- Klient side filter funksjonalitet og valg av listevisning

Bør være på plass for å få minst D

- Et RESTful API
- API-et må følge best praksis for API design med tanke på navngiving
- API endepunktene må bruke riktig HTTP verb
- Riktig bruk av responskoder
- Innloggingsfunksjon
- Klientside validering av skjema for opprettelse av artikkel (uten tredjeparts bibliotek)
- Serverside validering ved opprettelse av artikler (via mellomvare)
- CRUD operasjoner for fagartikler serverside og klientside
- Bruke ORM og Mongo til å lagre data

Bør være på plass for å få minst C

- Flere brukerroller
- Kontaktskjema med e-post håndtering (sending)
- Hensiktsmessige oppdeling av databasemodellene (1:N)

Bør være på plass for å få minst B

- Bildeopplasting for fagartikler
- Global errorhåndtering
- Tiltak mot CSRF og XSS
- Testing (et utvalg tester)
- Enkel API dokumentasjon via Postman
- Filtrere / paginere data serverside og klientside

Bør være på plass for å få A

- Logging av brukeradferd
- Superadmin rolle
- Statistikk basert på brukeradferd kun tilgjengelig for superadmin
- Eksport av statistikk i CSV / Excel
- Bruk av hensiktsmessig global statehåndtering
- God struktur i frontend og backend

Scorecard som gjør det enklere å sette poeng/sjekke krav ligger vedlagt.