Arbeidskrav 2 – JavaZone

Innleveringsfrist:	08.11.2024
Innleveringsformat:	Gruppe innlevering med lenke til Git-repository i PDF-format. PDF-filen leveres på wiseflow hvor dere kan opprette en gruppe og levere felles.
Emne ansvarlig	Dawood Ahmad (dawood.ahmad@gokstadakademiet.no)

Beskrivelse:

Du har fått oppdraget med å utvikle frontend-delen av en webapplikasjon for Javazone 2024, en av de mest prestisjefylte teknologikonferansene i verden. Sammen med en backend utvikler har dere ansvar for å lage et komplett system som håndterer foredragsholdere, foredrag og rom ved konferansen.

Backend-delen av applikasjonen er allerede utviklet (basert på en Express.js-server og en MySQL-database), og du skal nå bygge frontend i React. Applikasjonen skal bruke REST API-endepunktene som er tilgjengelige fra backend, som lar deg hente, oppdatere, legge til og slette informasjon.

Oppgavekrav:

1. **Du skal utvikle en applikasjon som kan:**
    a. Hente og vise data fra API-et:
        i. Hente og vise en liste over alle foredragsholdere.
        ii. Vise detaljer om en spesifikk foredragsholder når en bruker klikker på navnet deres.
        iii. Hente og vise en liste over alle foredrag.
        iv. Vise detaljer om et spesifikt foredrag, inkludert informasjon om rom og tidspunkt.
        v. Vise en oversikt over tilgjengelige rom og hvilke foredrag som er planlagt i hvert rom.
2. **CRUD-funksjonalitet (Create, Read, Update, Delete):**
    a. Brukeren skal kunne:
        i. Legge til nye foredragsholdere og foredrag via skjemaer.
        ii. Oppdatere informasjon om eksisterende foredragsholdere og foredrag.
        iii. Slette foredragsholdere og foredrag.
3. **Brukergrensesnitt:**
    a. Applikasjonen skal ha et enkelt, men oversiktlig design. Brukerne skal kunne navigere mellom ulike sider som viser foredragsholdere, foredrag og romoversikt.
    b. Det skal være en dedikert side for å vise alle rom og hvilke foredrag som holdes der.
4. **Innlogging:**

5. Implementer en enkel innloggingsside med brukernavn "**admin**" og passord "**admin**". Brukeren må være innlogget for å kunne legge til, oppdatere eller slette data.

API–dokumentasjon

Benytt test api verktøyet: https://crudcrud.com for å opprette og behandle data. Husk å slette cache fra nettleseren for å fornye den unike nøkkelen. Merk at data vil bli slettet ved fornyelse av nøkkelen. Her anbefales det å skrive et sett med requests som kjøres for hver gang du fornyer nøkkelen slik at du slipper å opprette ny data manuelt.

Her er en liste over API-endepunktene som skal brukes i applikasjonen:

Foredragsholdere:

Metode	Endepunkt	Beskrivelse	Parametere	Returnverdi
GET	**/api/speakers**	Hent en liste over alle foredragsholdere.	Ingen	Liste med objekter som representerer foredragsholdere, inkludert eventuelle tilknyttede foredrag.
GET	**/api/speakers/:id**	Hent detaljer om en spesifikk foredragsholder.	id (obligatorisk): ID-en til foredragsholderen.	Objekt med detaljer om foredragsholderen og en liste over deres foredrag (hvis noen).
POST	**/api/speakers**	Legg til en ny foredragsholder.	name (obligatorisk): Navn på foredragsholderen. bio (valgfritt): Kort biografi.	Objekt med detaljer om den nye foredragsholderen.
PUT	**/api/speakers/:id**	Oppdater en eksisterende foredragsholder.	id (obligatorisk): ID-en til foredragsholderen.	
DELETE	**/api/speakers/:id**	Slett en foredragsholder.	id (obligatorisk): ID-en til foredragsholderen.	Bekreftelse på at foredragsholderen er slettet.

Foredrag:

Metode	Endepunkt	Beskrivelse	Parametere	Returnverdi
GET	**/api/talks**	Hent en liste over alle foredrag.	Ingen	Liste med objekter som representerer foredrag, inkludert tilknyttede foredragsholdere og rom.

GET	**/api/talks/:id**	Hent detaljer om et spesifikt foredrag.	id (obligatorisk): ID-en til foredraget.	Objekt med detaljer om et spesifikt foredrag, inkludert informasjon om rom, tidspunkt og foredragsholder.
POST	**/api/talks**	Legg til et nytt foredrag.	title (obligatorisk): Tittel på foredraget. speakerId (obligatorisk): ID-en til foredragsholderen. roomId (valgfritt): ID-en til rommet hvor foredraget holdes. time (valgfritt): Tidspunkt for foredraget.	Objekt med detaljer om det nye foredraget, inkludert foredragsholder og rom hvis angitt.
PUT	**/api/talks/:id**	Oppdater informasjon om et foredrag.	id (obligatorisk): ID-en til foredraget. title, speakerId, roomId, time (valgfritt): Oppdaterte verdier	Oppdatert foredragobjekt med eventuelle nye tilknytninger til rom og foredragsholder
DELETE	**/api/talks/:id**	Slett et foredrag.	id (obligatorisk): ID-en til foredraget.	Bekreftelse på at foredraget er slettet.

Rom og tidspunkt:

Metode	Endepunkt	Beskrivelse	Parametere	Returnverdi
GET	**/api/rooms**	Hent en liste over alle rom.	Ingen	Liste med objekter som representerer tilgjengelige rom, inkludert eventuelle foredrag planlagt i hvert rom.
GET	**/api/rooms/:id**	Hent detaljer om et spesifikt rom.	id (obligatorisk): ID-en til rommet.	Objekt med detaljer om et spesifikt rom, inkludert en liste over foredrag som holdes der.
POST	**/api/rooms**	Legg til et nytt rom.	name (obligatorisk):	Objekt med detaljer om det nye rommet.

			Navn på rommet. capacity (valgfritt): Maksimal kapasitet for rommet.	
PUT	**/api/rooms/:id**	Oppdater informasjon om et eksisterende rom.	id (obligatorisk): ID-en til rommet. name (valgfritt): Oppdatert navn på rommet. capacity (valgfritt): Oppdatert kapasitet for rommet.	Oppdatert romobjekt med eventuelle nye verdier.
DELETE	**/api/rooms/:id**	Slett et rom.	id (obligatorisk): ID-en til rommet.	Bekreftelse på at rommet er slettet.

Antakelser:

Du SKAL anta at en foredragsholder kan opprettes uten å ha noen foredrag. Et rom kan opprettes uten å ha noen foredrag eller foredragsholdere. Et foredrag kan IKKE opprettes uten en foredragsholder.

Innleveringskrav:

En PDF-fil med lenke til ditt Git-repository må leveres på Github.
Sørg for ha repository private frem til arbeidskrav vurderingen er ferdig. Gitt at repository skal være private så trenger faglærer tilgang til repo-et.

Legg til brukeren til som 'collaborator' i repo: dawood11

Teamarbeid og Scrum i oppgaven

Denne oppgaven er ment å fullføres over 5 virkedager, og vi anbefaler at dere organiserer arbeidet i henhold til Scrum-metodikk. Scrum er en smidig prosjektstyringsmetode som ofte brukes i utviklingsprosjekter, og gir et godt rammeverk for effektivt teamarbeid.

Hvordan bruke Scrum på denne oppgaven:

1. Sprintplanlegging:

- Start med å planlegge oppgaven som en sprint på 5 dager. Del opp oppgaven i mindre arbeidsoppgaver (user stories). Eksempler på user stories kan være:
  - "Som bruker vil jeg kunne se en liste over alle foredragsholdere."
  - "Som bruker vil jeg kunne oppdatere informasjon om et foredrag."

2. Daglige standups:

- Hver dag bør teamet ha et kort møte (standup) der dere svarer på tre spørsmål:
  - Hva gjorde jeg i går?
  - Hva skal jeg gjøre i dag?
  - Er det noe som hindrer fremgangen min?

3. Tavleverktøy:

- Bruk et prosjektstyringsverktøy som Trello, Jira, eller GitHub Projects for å holde oversikt over oppgaver. Dette hjelper dere å visualisere arbeidsflyten og holde teamet oppdatert.

4. Samarbeid med Git og GitHub:

- Bruk branches: Når du jobber på en funksjon, bør du opprette en ny branch i Git. Dette gjør at alle kan jobbe parallelt uten å forstyrre hovedkoden.
- Pull requests: Når du har fullført arbeidet på en branch, send en pull request (PR) for å få den gjennomgått og deretter slått sammen med hovedkoden. Dette sikrer at endringer blir gjennomgått av andre i teamet før de legges til i hovedprosjektet.

5. Sprint Review og Retrospektiv:

- På slutten av sprinten (dag 5) bør dere ha en sprint review der dere demonstrerer hva som er blitt ferdigstilt. Deretter kan dere gjennomføre en retrospektiv for å reflektere over hva som fungerte bra og hva som kan forbedres til neste oppgave.

Læringsutbytte for oppgaven

Kunnskaper: Etter å ha fullført denne oppgaven, skal kandidaten:

- Ha forståelse for hvordan bygge fleksible og gjenbrukbare brukergrensesnitt med moderne JavaScript-rammeverk.
- Ha kunnskap om hvordan tilstandshåndtering og livssyklus administreres i applikasjoner.
- Ha innsikt i beste praksis for strukturering og organisering av kode for vedlikeholdbare applikasjoner.

Ferdigheter: Etter å ha fullført denne oppgaven, skal kandidaten:

- Kunne implementere og håndtere tilstand i komponentbaserte applikasjoner.
- Kunne dele data mellom ulike deler av applikasjonen på en effektiv måte.
- Kunne skrive og teste applikasjoner for å sikre funksjonalitet og stabilitet.
- Kunne feilsøke og forbedre applikasjoner basert på identifiserte problemer.

Generell kompetanse: Etter å ha fullført denne oppgaven, skal kandidaten:

- Kunne planlegge og gjennomføre en utviklingsoppgave både individuelt og i team, i samsvar med smidige arbeidsmetoder.
- Kunne bruke moderne utviklingsverktøy for samarbeid og deling av kode i et profesjonelt miljø.