NFUE23-25 | Emne 7 – Webteknologi | Eksamen H2024

Innledning

Velkommen til eksamensoppgaven i webteknologi! I denne oppgaven får du muligheten til å utvikle en viktig applikasjon for Sandefjord Lege Senter, som vil forbedre pasientopplevelsen ved å gjøre timebestilling enklere og mer tilgjengelig. Dette prosjektet gir deg ikke bare en sjanse til å anvende det du har lært om backend-utvikling med Node.js og Express, men også til å bidra til noe meningsfylt i helsevesenet.

Som utvikler vil du stå overfor reelle utfordringer og problemstillinger, og ved å bygge en fungerende løsning vil du lære verdifulle ferdigheter som vil være nyttige i din fremtidige karriere. Du vil arbeide med moderne teknologier, implementere autentisering, håndtere API-er og samhandle med en database. Dette prosjektet gir deg en flott anledning til å demonstrere din kreativitet og tekniske kunnskap.

Bakgrunnshistorie

Sandefjord Lege Senter er et moderne legekontor som ønsker å tilby pasientene sine en enklere måte å bestille timer på. De har sett behovet for å digitalisere timebestillingssystemet sitt, som i dag foregår via telefon eller oppmøte. Pasientene ønsker mer fleksibilitet, og Lege Senteret har derfor besluttet å utvikle en ny webapplikasjon for timebestilling.

Senteret har allerede kontaktet et eksternt byrå for å utvikle frontenden, som blir laget i **React**. Din oppgave som utvikler er å designe og implementere backend-løsningen. Backend-applikasjonen skal bygges ved hjelp av **Node.js** med **Express** som rammeverk. Applikasjonen vil koble seg til en database (f.eks. MySQL) for å håndtere brukerdata, timebestillinger, og administrasjon av leger og tilgjengelige tider.

Oppgavekrav

- Backend skal håndtere brukeregistrering og autentisering, inkludert beskyttelse av sensitive data som passord ved bruk av hashing (f.eks. bcrypt).
- En middelware for autentisering skal implementeres for å beskytte bestemte ruter som kun er tilgjengelige for autoriserte brukere.
- Systemet skal tillate pasienter å:
  - Se tilgjengelige legetimer
  - Bestille time
  - Avbestille eller endre en allerede bestilt time
- Legekontorets administrasjon skal kunne:
  - Legge til og administrere leger og deres tilgjengelighet
  - Se en oversikt over bestilte timer
- Det forventes at studentene lager en RESTful API-struktur og dokumenterer API-endepunktene.
- Frontend og design er ikke deres ansvar, men de må tilrettelegge API-et slik at tredjeparts frontend-team kan koble seg til og bruke backend-tjenestene.

Autentisering og brukerhåndtering

Metode	Endepunkt	Beskrivelse	Body	Respons

Metode	Endepunkt	Beskrivelse	Body	Respons
POST	`/api/register`	Registrerer en ny bruker (pasient).	{ navn, e-post, passord }	Bekreftelse på registrering
POST	`/api/login`	Logger inn en bruker og genererer en JWT-token.	{ e-post, passord }	JWT-token
POST	`/api/logout`	Logger ut brukeren og sletter tokenet.		Bekreftelse på utlogging

Timebestilling (for pasienter)

Metode	Endepunkt	Beskrivelse	Body	Respons
GET	`/api/appointments`	Henter alle tilgjengelige timer for timebestilling.		Liste over ledige timer
POST	`/api/appointments`	Bestiller en time for en pasient.	{ pasientId, legeId, dato, tid }	Bekreftelse på bestilling
PUT	`/api/appointments/:appointmentId`	Endrer en eksisterende time.	{ nyDato, nyTid }	Oppdatert timeinformasjon
DELETE	`/api/appointments/:appointmentId`	Kansellerer en bestilt time.		Bekreftelse på kansellering

Administrasjon (for legekontorets ansatte)

Metode	Endepunkt	Beskrivelse	Body	Respons
GET	`/api/admin/doctors`	Henter en liste over alle leger.		Liste over leger
POST	`/api/admin/doctors`	Legger til en ny lege i systemet.	{ navn, spesialisering, tilgjengelighet }	Bekreftelse på at legen er lagt til
PUT	`/api/admin/doctors/:doctorId`	Oppdaterer informasjonen om en lege.	{ tilgjengelighet }	Bekreftelse på oppdatering
DELETE	`/api/admin/doctors/:doctorId`	Fjerner en lege fra systemet.		Bekreftelse på at legen er fjernet

Metode	Endepunkt	Beskrivelse	Body	Respons
GET	`/api/admin/appointments`	Henter en oversikt over alle bestilte timer.		Liste over bestilte timer

Middleware

- Beskyttede ruter (f.eks. `/api/admin/*`, `/api/appointments`) skal sikres med autentisering middleware som sjekker JWT-token i headeren (f.eks. `Authorization: Bearer <token>`).

Innleveringsinstrukser

Rotmappenavn

Studentenes prosjekt skal legges i en rotmappe som er navngitt etter studentens fulle navn, med fornavn og etternavn adskilt med understrek (`_`). **Spesialtegn som æ, ø og å skal ikke brukes**. Dersom navn inneholder slike tegn, skal de erstattes på følgende måte:

- `æ` → `ae`
- `ø` → `o`
- `å` → `a`

Regler for rotmappenavn

- Dobbeltnavn skal separeres med understrek.
- Bindestreker i navn beholdes.
- Ikke bruk spesialtegn som æ, ø, å.

Eksempler

Navn	Mappenavn
Ola Nordmann	`Ola_Nordmann`
Anne Lise Larsen	`Anne_Lise_Larsen`
Kari-Anne Haug	`Kari-Anne_Haug`
Pål Ødegård	`Pal_Odegard`
Kåre Hansen	`Kare_Hansen`
Jan-Erik Østgaard	`Jan-Erik_Ostgaard`

Rotmappen skal inneholde følgende filer

1. **package.json**: Denne filen skal inneholde prosjektets metadata og avhengigheter.
2. **package-lock.json**: Generert av npm for å låse versjonene av avhengigheter.
3. **setup.sql**: SQL-skript som setter opp databasestrukturen brukt i prosjektet. (Viktig å testes grundig før innlevering, og vær bevist på scriptet ikke inneholder endringer av bruker eller tilganger)

4. **.env**: *Valgfri* konfigurasjonsfil som inneholder miljøvariabler, som databaseinnstillinger, men uten sensitive verdier (for eksempel passord eller API-nøkler).
5. **Prosjektfiler og -mapper**: Dette inkluderer alle filene og mappene som studenten selv velger å inkludere i prosjektet, som for eksempel kodestrukturen for backend-applikasjonen.
6. **Video**: En kort video hvor studenten beskriver prosjektet. Videoen skal fokusere på de viktigste momentene i løsningen, for eksempel:
    - Hvordan API-endepunktene fungerer.
    - Håndtering av autentisering og tilgangskontroll.
    - Hvordan databasen er satt opp og brukt i prosjektet.

MySQL Bruker

- Applikasjonen skal kobles til MySQL-serveren med brukernavnet `student`. Studenten skal bruke følgende passord for denne brukeren:
    - **Passord:** `9p;-n5:CGw9q`

Viktig

- **Ekskluder `node_modules`-mappen** fra innleveringen. Denne mappen kan gjenopprettes ved hjelp av `npm install`.

Videoformat

- **Varighet**: Maks 5 minutter.
- **Format**: MP4, MKV, eller et annet vanlig videoformat.
- **Navn på video**: `beskrivelse_video.mp4` (eller relevant filformat).

Kjøring av applikasjonen og oppsett av database

> ⚠ **Viktig:** *Studenten er selv ansvarlig for at applikasjonen kjører som tiltenkt. Sensorene vil ikke bruke tid på å rette opp i feil under vurdering av eksamen. Det er derfor viktig å teste applikasjonen grundig før innlevering.*

Forutsetninger

Før du kan kjøre applikasjonen, må du sørge for at du har installert følgende programvare:

- Node.js (versjon 14 eller høyere)
- MySQL (eller en kompatibel database)

1. Pakk ut prosjektet

Last ned oppgaven og pakk ut den zippede filen til ønsket mappe på datamaskinen.

2. Installere avhengigheter

Naviger til rotmappen av prosjektet og kjør følgende kommando for å installere nødvendige avhengigheter:

```
npm install
```

3. Oppsett av databasen

Første gang du kjører applikasjonen, må du sette opp databasen ved å kjøre SQL-skriptet. Følg disse trinnene:

1. Åpne terminalen.
2. Bruk følgende kommando for å logge inn på MySQL og samtidig kjøre `setup.sql`-skriptet:

   ```
   mysql -u student -p9p;-n5:CGw9q < path/to/setup.sql
   ```

   (Erstatt `path/to/setup.sql` med den faktiske stien til `setup.sql`-filen. Denne kommandoen vil logge inn med brukernavnet `student` og passordet `9p;-n5:CGw9q`, og deretter kjøre innholdet i `setup.sql`-filen, som oppretter databasen `eksamen_<studentens_full_navn>` og setter opp nødvendige tabeller.)

4. Konfigurer miljøvariabler (valgfritt)

Hvis du bruker en `.env`-fil, opprett en fil med navnet `.env` i rotmappen, og legg inn relevante miljøvariabler, for eksempel:

```
DB_HOST=localhost
DB_USER=student
DB_PASSWORD=9p;-n5:CGw9q
DB_NAME=eksamen_<studentens_full_navn>
```

5. Kjør applikasjonen

Start serveren med følgende kommando:

```
npm start
```

Applikasjonen skal nå være tilgjengelig på `http://localhost:3000`.

Vi ønsker deg lykke til med oppgaven! Husk å teste applikasjonen nøye og sørg for at alt fungerer som forventet. Vi ser frem til å se din kreative løsning! 🎓