Opgave: Full-stack portfolio projekt

Case beskrivelse:

I tidligere forløb har I arbejdet med backend (database og API) samt frontend (website) og integreret disse dele i et samlet system. I denne opgave skal I udvikle en komplet full-stack løsning fra bunden, hvor I både designer, udvikler og dokumenterer systemets forskellige lag. Opgaven skal fremlægges som en præsentation fredag d. 9. maj.

I vælger selv et emne og datasæt, som jeres løsning skal tage udgangspunkt i. Det kan eksempelvis være et system til håndtering af produkter, bookinger, brugere, anmeldelser eller noget helt andet. Vi har fundet lidt links der indeholder ideer til jeres software full-stack projekt (se nedenstående), vi opfordrer jer til at tage fat i noget udvikling til et område der interesserer jer. Nogle af linkene indeholder også URL'er eder leder jer hen til kodebaser i kan lade jer inspirere af:

https://www.fynd.academy/blog/full-stack-projects

https://www.geeksforgeeks.org/best-full-stack-project-ideas/

https://www.turing.com/blog/full-stack-project-ideas-for-software-developers

Der lægges vægt på, at løsningen struktureres efter principperne for god softwarearkitektur. Det betyder, at jeres kode skal følge SOLID-principperne og være skrevet efter Clean Code-principper. Jeres løsning skal derfor være opbygget i tydelige lag (fx dataadgang, forretningslogik og præsentation) og være modulær og testbar. Samtidig vil der være stort fokus på project structure, modulitet, scalerbarhed, integration mellem database, API og frontend. Der vil også blive afholdt et sparringsseminar i næste uge hvor vi har afsat tid til at hjælpe jer med at komme i mål med at understøtte de elementer, derudover vil I som sædvanlig have støtte fra akademiets team coaches og konsulenter. For at tilgængeliggøre forståelsen af jeres kode og jeres arbejdsproces skal i også løbende lave udførlige kommentarer i koden, en grundig ReadMe fil og instruktioner til hvordan man kører jeres software.

Vejledende specifikationer som jeres software projekt skal indeholde (MVP)

- En database med passende datamodeller og relationer.
- Et RESTful API til at håndtere forespørgsler mod data (CRUD-operationer).
- En frontend der kommunikerer med API'et og præsenterer data på en brugervenlig måde.
- En struktureret kodebase, hvor ansvar er adskilt efter arkitektoniske principper.
- Dokumentation af jeres arkitekturvalg og anvendte principper.

- Eksempler på unit tests eller integrationstests i minimum ét lag.

Der er mulighed for at kompensere i kompleksiteten på tværs af specifikationerne, snak med en team coach hvis i er i tvivl om jeres projekt lever op til de vejlende specifikationer.

Teknologier:

- .Net C#
- EntityFrameworkCore & Swashbukler
- PostgresSQL & Dbeaver
- TypeScript & React + Vite (frontend framworks er valgfrit)
- Docker & GitHub Actions

Foreslået Løsningsmetode (Vejledende step-by-step guide i kan lade jer guide af)

1. Opret en Web API projekt (der er template for det)
2. Opret og udfyld mapper, så som:
    - Models
    - DataContext
    - Repositories etc.
3. Lav end Seeder.cs file, som har til opgave at populate databasen efter man har lavet migration
    - Hvis I ikke selv kan finde data, kan I bruge cereal-dataen, som er vedhæftet til denne uges opgaveudlæg.
4. Lav et API til databasen ved at oprette og udfylde mapper så som:
    - Controllers
    - Interfaces
    - Dtos
    - Mappers
    - API'et skal understøtte CRUD-operationer på dataen i tabellerne.

5. Lav en frontend til API'et i en seperate projekt:
    - Brugeren skal kunne se billeder af cereal-produkterne.
    - Brugeren skal kunne klikke på et produkt og få vist yderligere information om produktet.
    - Brugeren skal kunne tilføje/fjerne et produkt til/fra kurven.
    - Brugeren skal kunne bekræfte købet, hvorefter databasen skal opdateres med brugerens oplysninger og køb.

I forhold til opbevaring af billederne til cereal-produkterne har I følgende muligheder:

- Gemme billederne i databasen som en base64-string.
- Gemme stierne til billederne i databasen.
- Bruge en cloud-service som Azure, AWS eller Google.

Autentifikation, UNIT testing, INTEGRATION testing, CI/DI piplies er elementer der sikrer robustheden af jeres software projekt. Vi opfordrer kraftigt til at I arbejder med branch handling så i får et softwareprojekt der er levende med ud på den anden side af akademiet som i kan bruge som et portfolio projekt der demonstrerer dine kodningskompetener. Vi skal dog gøre opmærksom at I i akademiets sidste uge ikke vil få stillet nogen ny opgave, men have tid til at genbesøge tidligere projekter eller elementer fra pensum inklusiv denne full-stack opgave.

Husk også at inddrag og videreføre elementer i synes var mest effektive til at understøtte jeres projektledelse og samarbejde fra full-stack opgaven i uge 6 & 7.