Full Stack Portfolioprojekt

Opgavebeskrivelse

Tidligere i forløbet har I arbejdet med backend (database og API) og 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.

I vælger selv det 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 opfordrer jer til at tage udgangspunkt i et område der interesserer jer!

Der lægges vægt på, at løsningen struktureres efter principperne for god softwarearkitektur. Det betyder, at jeres kode bør lade sig inspirere af **SOLID**-principperne, og **Clean Code**-principperne. Jeres løsning skal afspejle at i har reflekteret over de arkitektoniske valg og være opbygget i tydelige lag (fx. Dataadgang, forretningslogik og præsentation), og være modulær samt testbar.

Fordi projektet gerne skulle udmunde sig i et porteføljeprojekt, kan det samtidig være en god ide at understøtte en stærk projektstruktur, modularitet, skalérbarhed, samt integration til database, API og frontend. (Husk dog at det kan være vigtigt at opnå et minimal viable project (MVP), inden man begynder at understøtte et langsigtet vedligehold i projektet. Det kan sikre at man ikke over-engineerer ens projektet fra start).

Der vil blive afholdt et sparringsseminar mandag i næste uge, hvor vi hjælper hinanden med at komme i mål med ovenstående elementer, derudover vil I som sædvanlig få støtte fra akademiets teamcoaches og konsulenter.

Nedenfor har vi oplistet en række links, der indeholder ideer og inspiration til jeres projekt. Nogle links indeholder også URL'er, der leder jer hen til kodebaser I kan lade jer inspirere af:

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

- https://www.turing.com/blog/full-stack-project-ideas-for-software-developers
- https://dev.to/code_2/20-fullstack-project-ideas-that-will-land-you-a-job-in-2025-3h35

Afleveringsformat

Afleveringen til denne opgave består af følgende:

- En 15 minutters præsentation, hvor der lægges vægt på følgende:
  - Arkitekturvalg
  - Eventuelle designmønstre og kodeprincipper
  - Live demonstration
- Link til jeres Github-repository, der indeholder følgende:
  - Fungerende kildekode
  - En README.md-fil i dit projekt, hvor du forklarer:
    - Installationsvejledning
    - En kort teknisk dokumentation med UML-diagrammer og beskrivelse af implementerede designmønstre og arkitektur.

Vejledende specifikationer som jeres softwareprojekt 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 teamcoach, hvis I er i tvivl om hvorvidt jeres projekt lever op til de vejledende specifikationer.

Forslag til fremgangsmåde og opmærksomhedspunkter

Autentifikation, UNIT-testing, INTEGRATION-testing og CI/DI-pipelines, 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, og som kan bruge som et portfolioprojekt, der demonstrerer dine kodningskompetencer.

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

Brug gerne:

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

Vejledende step-by-step guide I kan lade jer inspirere af:

1. Opret et Web API-projekt (der er en template for det)
2. Opret og udfyld mapper, så som:
    - Models
    - DataContext
    - Repositories etc.
3. Lav en 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 et separat projekt:
    - Brugeren skal kunne se billeder af fx. 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.

Pensum og Ressourcer

[Text]

God arbejdslyst!