Studieguide

Inlämningsuppgift, FED23

Om lite drygt ett år sitter du där, på Frontend AB. Du är färdig utvecklare och är hypad till max!

Tyvärr har alla inte varit lika peppade, utan din föregångare gav upp mitt i projektet och sadlade om till snickare. (Han bygger numera datorchassin av trä.) Detta sätter dock Frontend AB i en besvärlig sits. De måste bygga klart projektet, men ingen av de gamla rävarna har tid. Det är här du kommer in!

Ditt uppdrag är att med hjälp av **testning** bygga klart en prototyp av "Studieguiden", en app för att hjälpa studenter att planera sina studier. Din föregångare har kommit en bit på väg, så du har kod att bygga vidare på.

1 User stories

1. Som en student vill jag att **veckans alla dagar ska visas**, så att jag kan välja fritt när jag vill göra mina uppgifter.
2. Som en student vill jag kunna **ta bort en todo item**, eftersom saker kan ändras.
3. Som en student vill jag kunna **ändra texten för en todo item**, så att jag kan uppdatera den om något nytt händer.
4. Som en student vill jag kunna **söka efter todo items** som innehåller en viss text, så att jag lätt kan se om en viss sak finns med i priolistan.
5. [VG] Som en student vill jag kunna **snooza en todo item** (dvs flytta till nästa dag), så att jag inte blir påmind om mina tidigare misslyckanden
6. [VG] Som en student vill jag kunna **starta nästa vecka** genom att alla todo items markeras som oavslutade, så att jag inte behöver lägga in alla todo items igen.
7. [VG] Som en student vill jag kunna **se en sammanställning** av totala antalet todos och hur många som inte är klara, så att jag får en överblick över min produktivitet. (Ex.: "15/20 klara")

För VG krävs minst 2 av 3 VG-kriterier implementerade.

---

2 Struktur

Utgå från koden här: https://github.com/ha-fed23-testning/study-planner
Lägg till nya komponenter och andra filer efter behov.

---

3 Testning

Du ska bygga appen med TDD. Det innebär: Red → Green → Refactor.
Du ska använda:

- **vitest** för enhetstest av fristående funktioner
- **Cypress** för komponenttest
- **Cypress** för e2e-test

Du behöver inte använda mocking i uppgiften.

8. Alla komponenter ska testas med komponenttest i Cypress. Utgå från de user stories som är aktuella för respektive komponent.
9. Hela appen ska testas med e2e-test i Cypress. Alla user stories ska testas med minst ett testscenario.
10. Funktioner i lösa JS-filer ska testas med enhetstest i vitest. (Undantag: store.js och date.js)

---

4 Demo

Appen ska **demonstreras för läraren** sista kursveckan.

---

5 Bedömning

Kursmålen, hämtade från kursplanen:

Efter genomförd kurs med godkänt resultat ska den studerande
Kunskap

1. Uppvisa konceptuell förståelse för var, när och varför man ska testa sin kod

Färdigheter

2. Kunna utföra enklare enhetstester
3. Kunna utföra enklare E2E tester

Kompetens (ansvar och självständighet)

4. Kunna testa sin kod

För **Godkänt** ska man ha genomfört kursen och nått alla målen ovan.

För **Väl Godkänt**:
"För att få betyget Väl Godkänt (VG) ska den studerande dels ha genomfört kursen och nått alla kursens läranderesultat, dels uppvisa en djupare förståelse för testning i rollen som frontend-utvecklare och dessutom ha att uppnått högre kvantitativa krav inom testning och uppvisa högre testtäckning"

"Högre kvantitativa krav" och "högre testtäckning" innebär att man ska testa utförligare. Till exempel kan man testa #1 genom att kontrollera att det visas 7 dagar. Men man kan testa utförligare genom att kontrollera att det är dagarna "Måndag" till och med "Söndag", i rätt ordning.

"Djupare förståelse" kan du demonstrera genom att skriva bättre test, som testar det som är relevant för projektets user story.

---

7 Frågor och svar

Q: Ska man använda riktiga datum?

A: Nej, det är en praktisk begränsning av prototypen för att den ska vara lättare att testa.

Q: Hur många testfall ska man göra för varje user story?
A: Så många som behövs. Olika user stories är olika komplicerade. Att kunna avgöra *vad som är relevant att testa* är en del av "konceptuell förståelse" och något som kommer tas med i bedömningen.

Q: Hur gör jag för att använda Zustand store tillsammans med Cypress?
A: Zustand har en inbyggd funktion för att komma åt store utanför komponenter. Exempel:

```
import { useStore } from './store.js'  // sökväg till din store-fil

it('exempel på test med Zustand', () => {
    // Läsa av innehållet i store
    const todos = useStore.getState().todos
    // todos är en lista med objekt, du kan använda .find för att hitta en
todo

    // Ändra i store - efter det kan du göra mount
    const newData = 'exempel'
    useStore.setState({ exempel: newData })
})
```

Q: Hur ska jag lämna in?
A: Lämna in en länk på LearnPoint. Lägg länken i en PDF och ladda upp.
För att kunna pusha den till ett eget repo, behöver du ta bort .git-mappen. Skriv "git init", skapa ett repo på github och koppla ihop dem.

Q: Hjälp, hur testar jag Item-komponenten med komponenttest?
A: Grundprincipen är att man testar det som är varje komponents ansvarsområde. Till exempel så är Item ansvarig för att visa texten på en "todo item". Då behöver man inte testa samma sak igen i Day- och Main-komponenterna.
Så som Item-komponenten är konstruerad så får den sin data via **props**. Den kommer inte att renderas igen när datan ändras i **store**. Det gör att det blir **omöjligt** att testa att **texten användaren ser ändras**. I stället kan man testa att **store uppdateras med rätt data**.

Exempel:

1. rendera en "todo item" med id=1337 och done=false
2. simulera ett klick på checkboxen
3. kontrollera att todo med med 1337 i Zustand store har ändrats så att done===true

Q: Panik! Det går inte att starta E2E-testen, det funkade ju igår?!?
A: Du behöver nog starta din lokala utvecklingsserver: `npm run dev`

.