Inlämningsuppgift 1 (U/G)

Deadline: 26.09.21

Webb-applikationen SameTaste

Ni ska koda en enklare webbaserad social media app som bygger på the Met's API (https://metmuseum.github.io/). Ni ska använda både Met's API:n och SameTaste API:n (beskrivs i ett annat dokument). Appen finns att se i videon.

Uppgiften lämnas in:

- På Canvas som en zip-fil. Filerna i zippen ska kunna laddas upp på en webbserver och allt ska fungera utan vidare.
- Som en URL (appen ska alltså finnas tillgänglig på webben). Ni kan publicera den var ni vill men det finns en webbsupport-kod för var och en av er som ni kan använda för att skapa en egen .se-domän, om ni inte redan har en. Webbhotellet är gratis första året. Skriv ett mail till mig om ni vill ha en kod.

Beskrivning av appen

Se videon för att få en bättre känsla för beskrivningen.

För att klara uppgiften ska ni koda en webb-baserad app med följande funktionalitet:

A. Huvudanvändaren (HA, Erik i videon) är den som är inloggad på appen. För att förenkla uppgiften kommer vi inte att skapa en login-funktionalitet. HA deklareras i koden (förslagsvis med const mainUserID = 0;). Ni ska var och en av er välja en HA (id) från listan i slutet av dokumentet. Ni kommer att se vilken som är menad för er.

B. När appen laddas ser HA en lista av verk från Mets samling, närmare bestämt de från avdelningen "European Paintings"som har ordet "snow" i någon av texterna. Det är verken som man får om man skickar följande GET-förfrågan: https://collectionapi.metmuseum.org/public/collection/v1/search?departmentId=11&q=snow

   A. Förfrågan ovan returnerar en lista på id:s som var och en representerar en målning. För att få information om målningen måste vi skicka en ny förfrågan. Information om hur den ska formuleras och svaret från den finns att läsa här. Nycklarna vi är intresserade av är objectID (Mets id för denna bild), primaryImageSmall (url:et till själva jpg-filen), title (bildens titel, som ska användas i texten), artistDisplayName (används i texten också)

C. HA kan välja sina favoritverk från listan. Det gör hen genom att klicka på add-knappen. Om verket är ett av favoriterna ska detta markeras (i videon en grön kant). Ett favoritverk ska ha en remove-knapp. Genom att klicka på den så tar HA bort verket som en av favoriterna.

D. Det finns ett maximalt antal favoriter som en HA kan välja. Denna siffra bestäms av API:n, publiceras inte och kan ändras när som helst. Det som dock händer om ni försöker lägga till för många favoriter hos en användare är att ni får en 409 response (conflict). Detta måste ni kontrollera så att appen ger användaren lämplig feedback. Ni måste designa feedback:en.

E. När appen laddas så är HA den "valda" (selected) användaren och hens namn är markerat (svart bakgrund)

F. När man klickar på någon av de andra användarna (inte HA) syns hens favoriter. Notera att det är endast hens favoriter som visas, inte alla verk (till skillnad från HA, där alla verk visas så hen kan välja favoriter).

G. Huvudanvändaren (alltså användaren som är inloggad, fast vi ska inte ha någon inloggning) kan endast ändra sina favoriter, inte de andras. Därför syns inte några add- eller remove-knappar när man visar andra användares verk.

H. Verken som HA och den valda användaren båda har som favoriter markeras på ett annat sätt (i videon en röd kant runt ramen).

I. Bredvid ATESTERs namn visas inom brackets hur många favoriter hen har valt (precis som för HA). Efter brackets visas, inom vanliga parenteser, hur många av ATESTERs favoritverk som är samma som HAs. Detsamma gäller för alla namn i listan. Det är inte meningsfullt att visa hur många av HAs favoriter är samma som HAs favoriter, därav inga vanliga parenteser bredvid HAs namn.

Andra krav

1. **All kommunikation med servrarna måste lösas med async och await**
   Det är dock ok om någon enstaka Promise fixas med .then() istället för async + await.
   Som vanligt bör ni identifiera vilka funktioner ni behöver (ev. argument, ev. returvärde, vad de ska göra) innan ni börjar koda.
2. Bilderna ändras ju inte, det är bara vilka som är favoriter som ändras. Det är därför ett krav att **informationen om bilderna ska endast fetchas en gång**, när sidan laddas upp. Sidan ska endast behövas laddas upp en gång.
   1. Under inga omständigheter får bilderna tas bort och läggas till när användaren adderar eller tar bort favoriter.
   2. Det är däremot ok att ta bort och lägga till användarna. Det går väldigt fort.
3. **Kontinuerlig uppdatering**. Uppdatering av informationen sker vid två olika tillfällen:
   1. Varje gång HA lägger till eller tar bort en favorit. Notera att API:n svarar med en uppdaterad lista på alla users. När svaret kommer ska appen uppdatera alla siffror inom brackets och parenteser.
   2. Varje 30:e sekund. Då ska appen skicka en förfrågan till SameTaste för att hämta den uppdaterade informationen om användarnas favoriter. Siffrorna inom brackets och parentes ska uppdateras när datan har kommit in.
4. **Feedback vid uppdatering av users.**
5. **Feedback vid hämtning av info om bilderna (endast när de inte finns i localStorage)**. Se videon. Se nedan för mer info.
6. **Feedback vid add/remove favorit**. När användaren lägger till eller tar bort en favorit skickas det en förfrågan till SameTaste. Under tiden som förfrågan sker ska användaren få feedback.
7. **Feedback vid regelbundna uppdateringen**. Under tiden förfrågan skickas (var 30:e sekund) ska detta markeras med någon form av feedback. I videon visas en sådär form av feedback som dock accepteras som lösning. Ett mycket bättre alternativ vore någon form av timer som informerar om hur lång tid det är kvar till nästa uppdatering samt en feedback när uppdateringen sker som inte täcker alla namn. Det är upp till dig om du vill designa och koda en sådan istället. Det finns dock inte möjlighet att få VG i denna uppgift.
   1. Vi kommer inte i denna kurs att lära oss programmera WebSockets, vilket är vad man borde använda i en app som denna, istället för att skicka förfrågningar titt som tätt.
8. För att appen ska starta upp snabbare ska ni **spara information om bilderna i localStorage**:
   1. Denna information ska sparas som en array av objekt där varje objekt har nycklarna:
      1. objectID (Mets id för denna bild)
      2. primaryImageSmall (url:et till själva jpg-filen)
      3. title (bildens titel, som ska användas i texten)
      4. artistDisplayName (konstnärens namn, som ska användas i texten)
   2. Första gången appen startas på en dator+webbläsare finns det ingen information om bilderna i localStorage. Informationen ska hämtas och sparas som beskrivet ovan.
   3. Alla de andra gångerna appen startas ska datan från localStorage användas.
   4. TIPS när ni ska testa: Man kan rensa allt som finns i localStorage genom att på konsolen skriva localStorage.clear(), så att man startar appen som första gången.
9. **Sortering**
   1. Användarna är sorterade enligt namn i stigande ordning.
   2. Verken ska vara sorterade enligt konstnärens förnamn i stigande ordning.
10. **Grafiken**
    1. Appen ska fungera och se bra ut för viewer-width över 1000px. Om ni vill kan ni tvinga "wrappern" till den storleken och:
       1. Om viewern är större => centrera wrappern
       2. Om viewern är mindre => scrollbars.
       3. Vi kräver alltså inte responsiv design i denna uppgift.

2. I videon presenteras ett förslag på hur appen kan se ut. Ni kan följa den eller designa en egen lösning. Man ska kunna se minst 8 hela verk på en 1000x700px skärm (ungefär som på bilderna). Videon visar ungefär var ribban ligger vad gäller utseende för att bli godkänd.
3. Namnen och verken ska ha var sin egen scroll (kanske inte behövs för användarna, beroende på hur ni designar den delen).
4. Det får inte förekomma några andra scrolls än just de två ovannämnda.
5. I appen representeras verken med själva tavlan och en text.
   A. Texten består av "*Title*, by *Author*".
   B. Den vita ramen runt tavlan måste vara fyrkantig även om tavlorna är sällan det.
   C. Det måste alltid finnas minst 10px vit ram runt bilden på tavlan.

Andra krav

- Korrekt indentering av all kod
- Relevanta namn på variabler (något som informerar om vad variabeln innehåller)

Users som finns på SameTaste DB:n

Du förstår säkert vem du är i listan nedan. Den visar användarna (sorterade på efternamn) som finns på SameTaste DB:n. Ni ska välja er själva som huvudanvändare i appen som ni utvecklar. Försök att inte ta fel där så ni inte jobbar mot samma user på DB:n. Det kan bli konstiga resultat.

Lina, id: 0
Ludvig, id: 1
Olivia, id: 2
Alice, id: 3
MatildaE, id: 4
Pranvera, id: 5
Isak, id: 6
Ivan, id: 7
Rasmus, id: 8
Elsa, id: 9
Clara Marie, id: 10
Jonna, id: 11
MatildaN, id: 12
Bea, id: 13
Kim, id: 14
Melinda, id: 15
Sandra, id: 16
Niklas, id: 17
Edit, id: 18
Mehmet, id: 19
Sally, id: 20
Alexander, id: 21
Mercie, id: 22

Tester

Ni kommer att behöva testa om ändringar i andra användares favoriter blir uppdaterade på rätt sätt i er app. Säg att Josefine vill testa om hennes app uppdateras korrekt när Hulda väljer (eller väljer bort) ett specifik verk. För att ni inte ska behöva be varandra om att ändra era val så har jag skapat en test-användare (ATESTER).

ATESTER har som favoriter alla verk av:
Aert van der Neer, Alfred Sisley, Barend Cornelis Koekkoek, Edouard Manet, Edward Lear, El Greco (Domenikos Theotokopoulos)