Dokumentation slutprojekt - Musikstreamingtjänst

Johannes Larsson Lewin
Webbserverprogrammering

1. Utfört arbete

1.1 Frontend

Projektets frontend är skriven i HTML, CSS och JavaScript. Jag har skapat följande sidor och funktioner:

- Inloggning och registrering: Formulär för att logga in och skapa konto, med validering och felhantering.
- Startsida: En modulär layout där användaren kan flytta och stänga av element. Detta görs med drag-and-drop funktionalitet implementerad med JavaScript. Det här var nog det mest främmande för mig och jag behövde lära mig mycket nytt.
- Sökfunktion: Sökruta kopplad till YouTubes API.
- Spellistor: Sida där användare kan skapa och se alla sina spellistor samt spela upp dem. Jag valde att flytta bitar av det här till min huvudsida för att göra det lättare för användare.
- Musikspelare: Egen musikspelare med funktioner för uppspelning, kö och låtinformation. Jag har fokuserat på att göra den här så modulär som möjligt och koden kan användas till andra projekt.
- Kommunikation: Jag har mycket kommunikation mellan mina skript och webbsidor, både i backend med databas och frontend mellan sidor. Jag har kommunicerat med flera olika tekniker i min fronten; Custom Events, LocalStorage och även med vanliga variabler.

1.2 Backend

Backenden är skriven i Node.js med Express. Jag har implementerat följande huvudfunktioner:

- Autentisering: Säker inloggning och registrering med krypterade lösenord (bcrypt) och sessionshantering (express-session).
- YouTube-integration: Kommunikation med YouTubes API för sökningar och hämtning av låtinformation.
- Ljudströmning: Strömning av ljud från YouTube med hjälp av ytdl-core. Skyddat bakom inloggning.
- Databashantering: Koppling till MySQL-databas för lagring av användare, spellistor och låtar.

Exempel på Express-route för ljudströmning:

```
app.get('/audio/:id', (req, res) => {
        if (!req.session.user)
        return res.status(401);

        const url = `http://www.youtube.com/watch?v=${req.params.id}`;

        ytdl(url, { filter: 'audioonly' })
        .pipe(res);
})
```

1.3 Databas

Jag har skapat en MySQL-databas med följande tabeller:

- Users: Lagrar användarinformation som användarnamn, krypterat lösenord, email och registreringsdatum.
- Playlists: Lagrar information om spellistor som namn, beskrivning, ägare (user_id) och publiceringsstatus.
- Playlist_songs: Kopplingstabell mellan spellistor och låtar. Lagrar låtdata och datum tillagd.

Tabellerna är normaliserade för att undvika duplicerad data och möjliggöra effektiva sökningar.

1.4 Tidslinje

Vecka 1
Under denna vecka gjordes en grov ritning av indexsidan i HTML och CSS med placeholders. Layout och stil på startsidan utvecklades vidare.

Vecka 2
Fokus låg på frontendutveckling för startsidan med JavaScript, inklusive hantering av sidomenyn och integration med YouTubes API för att få fram sökresultat. Det gjordes även många stiländringar och vidareutveckling av startsidan med placeholders. Design av söksidan påbörjades och kommunikation med backend och API för att visa sökresultat implementerades.

Vecka 3
Arbetet fortsatte med vidareutveckling av styling på söksidan. Små ändringar gjordes i api.js, samt stil- och funktionsutveckling på söksidan. Utveckling av server, autentisering och spellista påbörjades, inklusive streaming av ljud, registrering, inloggning och initial spellista-logik. Kommunikation mellan search.js och player.js för uppspelning av ljud implementerades, vilket gjorde det möjligt att söka, välja och spela låtar.

Vecka 4
Denna vecka fokuserade på design av spellistasidan och implementering av placeholder-kod för att se upplägget. Funktioner för att skapa, lägga till låtar och se spellistor serverside utvecklades, och testfunktioner på klient implementerades. Styling av clientside-kod för spellistor gjordes, tillsammans med funktionalitet och uppspelning av spellistor med kommunikation via localStorage. Spellistadelen flyttades till startsidan, placeholder-kod ersattes med riktiga låtar som trendar på startsidan, och databasen kopplades ihop med backend. Dokumentationen skrevs också under denna period.

Planeringen har fungerat bra men flera moment som jag hade dedikerat veckor till i planeringen utfördes parrallelt för att spara på tid.

2. Ändringar från ursprunglig planering

- Jag hann inte implementera kommentarsystemet likt SoundCloud. Prioriterade andra kärnfunktioner.
- Podcast-delen hann inte med. Kan läggas till i framtida vidareutveckling.
- Lade till Playlists på startsidan, samt några extra knappar i musikspelaren

3. Testning

3.1 Tester i olika webbläsare

Jag har testat webbplatsen i Chrome, Firefox, Safari och Edge. Layouten och funktionerna fungerade som förväntat i samtliga webbläsare. Inga större skillnader upptäcktes. En sak som inte fungerar perfekt i äldre webbläsare (typ Internet Explorer) är backdrop-filter. Och någon annan css funktion som t.ex. translate. Det här påverkar dock inte användning av webbsidan. Webbsidan är också mobilanpassad och det går bra att navigera och använda webbsidan på mobilen.

3.2 Användartester

Jag lät 4 personer i målgruppen testa sidan och ge feedback. Överlag positiv respons på design och funktionalitet. Några förbättringsförslag:

- Tydligare felmeddelanden vid inloggning/registrering
- Möjlighet att redigera spellistor efter skapande
- Fler filteralternativ i sökfunktionen
- Bättre generell styling skulle va bra

Jag hade planerat filteralternativ och egentligen skulle hela api.js behöva skrivas om för att minska koduppprepning. inloggning / registrering har jag inte lagt ner mycket på då jag prioriterade andra (mer komplicerade) delar av webbsidan; Det här sade jag också i min showcase-video.

Jag fick feedback i olika delar av utvecklingen i sidan, och en som jag har lyssnat mycket på är styling. Även fast det kanske inte är jätteviktigt för betygskriterierna vill jag ändå ha en snygg sida, så jag har visat upp sidan under utvecklingsprocessen. Ett exempel på feedback jag lyssnat på är att jag ett tag hade överdrivet mycket rounding och align center på element. Det här är förbättrat nu.

3.3 Säkerhetstester

- Testat SQLi genom att skicka in specialtecken i sökningar och formulär. Inga sårbarhet funna..
- Verifierat att lösenord lagras krypterat i databasen och aldrig skickas i klartext.

- Säkerställt att endast inloggade användare kan strömma ljud och komma åt skyddade routes.

4. Lagar och säkerhet

4.1 Upphovsrätt och GDPR

Allt innehåll strömmas direkt från YouTube och lagras inte på egna servrar, för att respektera upphovsrätt.

Jag använder mig av många ikoner på webbsidan; det har jag hämtat från bootstrap som tillåter fri användning i alla syften. Utöver det så är allt material på webbsidan skapad själv. Det enda jag sparar från användare i databasen är användarnamn och lösenord. Vid en potentiell läcka så skulle bara användarnamn och ett krypterat lösenord komma ut.

4.2 Säkerhetsåtgärder

- Jag använder bcrypt för säker lösenord lagring.
- Skyddar routes bakom autentisering med express-session. Det här är inte nödvänligt för säkerhet men görs för att undvika rate limits från youtube. Om en aktör skulle få för sig att skicka massa förfrågningar så skulle jag i alla fall ha deras användarnamn så att jag kan ta bort deras konto. En sak att utveckla kan vara att sätta tidsbegränsningar på olika requests för att låta alla användare söka och streama musik. Om massa användare skulle försöka använda min tjänst idag så skulle youtube säga ifrån då jag använder mig av ett globalt service konto för mina requests istället för att använda användarens egna youtube konto.
- Servicekontot gör dock att det blir säkrare för användare, då de inte behöver ge webbsidan tillgång till deras Google- konto.
- Jag validerar och sanerar all användargenererad data innan databaslagring eller visning.
- Alla npm paket som jag har använt mig av är välkända och populära.

5. Kodgenomgång

5.1 Viktiga funktioner

Drag-and-drop för modulär startsida:

- Lyssnar på events som dragstart, dragover, drop osv.
- Ändrar ordning på element i DOM baserat på användarens drag-and-drop.
- Sparar ordning och status i localStorage.

Sökfunktion med YouTube-integration:

- Skickar sökterm till server med fetch-anrop.
- Server kommunicerar med YouTubes Data API för att hämta sökresultat.
- Resultat visas dynamiskt på sidan med data från YouTube och egen databas.

Musikspelare:

- Hämtar strömmande ljud från server med HTML5 Audio.
- Styr uppspelning, kö och visar låtinfo.
- Uppdaterar UI i realtid baserat på uppspelningsstatus med hjälp av eventlisteners.

5.2 API:er och bibliotek

- YouTube Data API v3 - Sökningar och hämtning av låtinfo
- ytdl-core - Strömning av ljud från YouTube
- Express - Web framework för Node.js
- express-session - Sessionshantering för autentisering
- bcrypt - Kryptering av lösenord
- MySQL2 - MySQL-klient för Node.js

Ett exempel på kommunikation via frontend är search.js och player.js.

search.js
Läser av knapptryck i sökfält, skickar innehåll till servern och får tillbaka sökresultat i en lista som är baserat på innehållet. Efter det så loopar vi igenom listan och genererar ut varje del i html DOM samtidigt som en eventlistener läggs till som lyssnar efter mustryck. När användaren klickar på en låt så skapar vi ett custom event med hela låtobjektet som detalj i meddelandet.

player.js
Här lyssnar vi efter flera custom events från search.js, men ett är song_added. Om de eventet utlöses så tar vi detaljerna från eventet och anropar play_song med informationen om låten, som sedan spelar musiken och uppdaterar låtinformation på sidan.

I kommunikation mellan webbsidor (som t.ex. mellan startsidan och search.html) så används localStorage. Det här är eftersom events inte kan upptäckas mellan webbsidor och eftersom vi vill minimera koden som behöver köras på servern.

5.3 Kodförbättring

En vidareutveckling i anknytning till kommunikation kan vara att börja använda en “shared worker” för ljuduppspelning. En shared worker kan kommunicera mellan sidor och det skulle vara bra för att undvika duplicering av ljuduppspelning. Det vi gör i dagsläget för att undvika det är att bara spela ljud från search.html, men det kräver att search alltid behöver vara öppet i webbläsaren vilket kan vara jobbigt. En annan fix till det här problemet kan vara att ta bort alla andra sidor utom index och hantera html generering med javascript, men den här lösningen leder till överkomplicerad kod och mer som kan gå fel; allt blir mindre modulärt.

En annan ändring som jag skulle göra nu i efterhand är hur vi hanterar vår que popup. Den som finns nu uppdateras inte automatiskt. En lösning kan vara att sätta funktionen på en timer, men det vill man undvika som standard. En bättre lösning är att ändra hur vi hanterar

localstorage och spara vår que dit direkt en ändring har skett, och sedan lyssna efter ändringar i localstorage och uppdatera que:n efter det. En ändring som krävs då är att kommunikation mellan sidor med localstorage behöver ändras.

6. Utvärdering

6.1 Problem och lösningar under projektets gång

- Problem med autentisering och sessionshantering. Löstes genom att noga följa dokumentation och säkra exempel. Var även krångligt med att få åtkomst till bilder som sparades på servern.
- Utmaningar med att integrera YouTubes API. Innan jag hittade youtubes kodexepel var det skrämmande då jag aldrig har använt api:er som har varit så komplexa tidigare.

6.2 Vidareutveckling

- Lägga till stöd för podcasts enligt ursprunglig plan.

Det här görs lätt med sökfilter.

- Utveckla ett rekommendationssystem baserat på lyssningshistorik.

En annan utveckling är en smart shuffle som börjar när alla låtar i que:n har spelats. Man kan göra en enkelt version genom att söka på låtar från artisten som man senast lyssnat på och ignorera alla låtar som matchar last_three arrayen.

- Möjlighet att dela spellistor med andra användare.
- Förbättra gränssnittet ytterligare med fler animationer och micro-interaktioner.

Det här är en generell punkt; men ett exempel kan vara i search när man klickar på de 3 små knapparna för att få upp more menyn.

6.3 Slutsatser och lärdomar

Detta projekt har gett mig värdefulla erfarenheter i fullstack-utveckling. Några viktiga lärdomar:

- Vikten av noggrann planering och tidsuppskattning.
- Fördelarna med att dela upp kod i modulära komponenter.
- Att testa tidigt och ofta för att fånga buggar och användbarhetsproblem.

Jag är stolt över slutresultatet och kommer använda delar av koden i kommande projekt.

7. Betygsvärdering

Baserat på projektets omfattning, tekniska komplexitet och genomförande anser jag att jag har uppnått kraven för betyget A. Jag har använt mig av många olika tekniker och jag har bra strukturerad kod som gör många saker i samspel.