#### TITTEL & SAMMENDRAG

**Tittel:** ReviewHub: En webapplikasjon for anmeldelser av filmer.

**Sammendrag:** ReviewHub er en webapplikasjon designet for å tillate brukere å se, legge til, og interagere med filmanmeldelser på en brukervennlig plattform. Prosjektet kombinerer moderne webteknologier for å skape en tilgjengelig og interaktiv opplevelse. Vårt mål var å tilby en plattform der filmentusiaster kan dele sine opplevelser og meninger om filmer gjennom anmeldelser og kommentarer. Ved bruk av HTML, CSS, JavaScript på frontend og ASP.NET Core med en MySQL database på backend, har vi sikret en sømløs brukeropplevelse med sikkerhetsmekanismer gjennom bruk av JSON web token-autentisering. Prosjektet har oppnådd målene om brukervennlighet, samt tilgjengelighet og sikkerhet, noe som reflekteres i systemets evne til å håndtere brukerinteraksjoner og data på en effektiv måte.

#### INNLEDNING

**Prosjektbeskrivelse:** ReviewHub er en nettbasert plattform hvor filminteresserte kan lese og dele anmeldelser av filmer. Plattformen tilbyr et samfunn hvor brukere kan bidra og engasjere seg i innhold gjennom funksjoner som å legge til nye filmer med relevant informajson inkludert bildeopplastning(MoviePoster), skrive anmeldelser av filmer inkludert mulighet for bildeopplastning(ReviewPoster), og kommentere på andres anmeldelser.

Alle brukere, uavhengig av om de er innlogget eller ikke, kan se alle tilgjengelige filmer med deres rating og annen relevant informasjon, samt lese anmeldelser og kommentarer. Det er kun funksjoner som involverer interaksjon med systemet som krever innlogging og autentisering med JWT-token, basert på brukernavn og passord.

**Relevans og mål:** I dagens digitale tidsalder er interaksjon og engasjement blant brukere viktige aspekter ved enhver online plattform. Det er mange nettsteder og plattformer som tilbyr muligheten til å vurdere og gi tilbakemelding på filmer og TV-serier, men det er få som tillater brukerne å uttrykke sine meninger gjennom kommentarer på anmeldelser.

Denne funksjonen er på ingen måte original, men fraværende hos de store og etablerte nettstedene som «IMDb» og «Rotten Tomatoes».

Dette er en viktig utfordring som ReviewHub ønsker å takle på en innovativ, brukervennlig og effektiv måte. Ved å tilby et rom for å diskutere og kommentere anmeldelser, går ReviewHub utover det å bare å gi brukerne muligheten til å gi en tommel opp eller tommel ned på en vurdering.

Det var også et mål at brukerne selv skulle ha kontroll over sin digitale identitet. Registreringsprosessen ble derfor implementert på en slik måte at fornavn og etternavn er valgfrie felter. Dette valget reflekterer ønsket om å gi brukerne autonomi og trygghet når de deltar i fellesskapet på ReviewHub.

Ved å tilby kommentarfunksjon, sammen med muligheten for anonymitet, håper ReviewHub ikke bare å tilby en plattform for å vurdere og dele meninger om filmer, men også å dyrke et dynamisk fellesskap hvor lidenskapen for film kan blomstre gjennom åpne diskusjoner og interaksjoner.

**Målgruppe:** ReviewHub er for filmelskere av alle slag. Plattformen er designet for å fange interessen til alt fra kultfilm-entusiaster, til de som like gjerne setter på den nyeste blockbuster-filmen.

**Prosjektmål:** Prosjektets mål inkluderte å skape en enkel og sikker registreringsprosess, muligheten for brukerne å engasjere seg aktivt gjennom anmeldelser og diskusjoner, og å sikre at plattformen var tilgjengelig på tvers av forskjellige enheter og plattformer.

#### METODE

**Utviklingsprosess:** Utviklingen av ReviewHub fulgte en iterativ tilnærming, hvor vi startet med en grunnleggende versjon og gradvis la til funksjonalitet basert på brukertesting(oss selv) og identifiserte behov.

**Frontend:**Utviklingen av frontend brukergrensesnitt og design ble gjennomført ved bruk av HTML, CSS, og JavaScript for å skape en interaktiv og brukervennlig brukeropplevelse. Vi brukte moderne webteknologier for et responsivt design som tilpasser seg ulike skjermstørrelser. JavaScript ble intensivt brukt for å håndtere brukerinteraksjoner og dynamisk innhold. En vesentlig del av frontend-utviklingen var integrasjonen med backend via fetch-metoden, for å sende og motta data asynkront. Dette gjorde det mulig for brukerne å utføre handlinger som å legge til og vurdere filmer uten å måtte laste siden på nytt, noe som forbedrer brukeropplevelsen betydelig.

**Backend-arkitektur:**

Backend-delen av ReviewHub ble utviklet som en kontrollerbasert API ved å bruke ASP.NET Core, etter en lagdelt arkitektur for å sikre modulær kode:

* **Kontrollerlag:** Dette laget håndterte direkte brukerinteraksjoner gjennom HTTP-forespørsler. Det fungerte som en bro mellom brukergrensesnittet og forretningslogikken ved å delegere oppgaver til tjenestelaget og sende svar tilbake til klienten.
* **Servicelag:** Her lå forretningslogikken for applikasjonen, inkludert brukerautentisering og håndtering av filmer, anmeldelser og kommentarer. Tjenestelaget utførte funksjonene som kreves for å administrere datastrømmer og sikre at forretningsregler og prosesser ble korrekt implementert.
* **Repositorylag:** Ansvarlig for all direkte interaksjon med databasen gjennom Entity Framework. Dette ga en abstraksjon av datalaget og forenklet tilgang til databasen uten behov for å håndtere databaseforbindelser eller SQL-spørringer direkte.

**Databaseintegrasjon og migrering:**

ReviewHub bruker MySQL-database gjennom Entity Framework Core for å lagre og administrere data:

Vi definerte nødvendige enheter i API’et, som representerer de ulike dataobjektene i systemet, inkludert brukere, filmer, anmeldelser og kommentarer. Deretter utførte vi migreringer ved hjelp av Entity Framework CLI-verktøy, som genererte de tilsvarende databasetabellene basert på enhetene og deres relasjoner.

**Feilhåndtering og validering:**

For å sikre robust feilhåndtering på globalt nivå, implementerte vi en GlobalExceptionMiddleware som fanget opp unntak som ikke ble håndtert andre steder i applikasjonen og ga en brukervennlig feilmelding tilbake til klienten. Vi brukte også FluentValidation for å validere inndata både under registrering og pålogging.

**Autentiseringssystem:**

Vi utviklet en autentiseringsflyt ved å bruke JSON Web Tokens (JWT) for å administrere økter og sikre API-forespørsler:

Under pålogging blir brukerens legitimasjon validert mot databasen, og passord sikres gjennom hashing ved hjelp av Bcrypt.

Ved vellykket autentisering genererer backend-serveren et JWT-token, som sendes tilbake til klienten og må inkluderes i alle påfølgende API-kall som krever autentisering.

// VET IKKE HVA VI SKAL SI HER, VI HAR JO JOBBET ALL OVER THE PLACE

**Samarbeid og arbeidsdeling:** Planen var at Robin skulle fokusere på backend-utvikling, databaseintegrasjoner og JWT-token autentisering, mens Sakaria skulle fokusere på frontend-design og integrasjonen med backend, samt backend-utvikling.

Det har vært mye frem og tilbake, og vi har begge jobbet både på frontend og backend. Vi har hatt kontinuerlig kontakt i form av ukentlige Teams-møter og hyppig bruk av GitHub.

**RESULTATER**

I løpet av prosjektet har vi oppnådd flere resultater som møter de opprinnelige målene vi satte. Disse resultatene inkluderer:

**Autentiseringssystem:** Vårt autentiseringssystem, som anvender JSON Web Tokens (JWT), har spilt en kritisk rolle i håndteringen av brukersesjoner og sikringen av API-forespørsler. Systemet har effektivt sikret brukerdata ved hjelp av hashede passord og JWT-er for å vedlikeholde brukerens sesjon på klienten gjennom localStorage. Dette har tillatt sikker innlogging og kontinuerlig håndtering av brukersesjoner. Det må imidlertid påpekes at selv om JWT-er fungerer sømløst fra frontend til backend, representerer lagringen av disse tokenene i localStorage en potensiell sikkerhetsrisiko. Dette skyldes at tokens lagret i localStorage er tilgjengelige for skript kjørende i nettleseren, noe som kan utgjøre en risiko for cross-site scripting (XSS) angrep. Vi planlegger å forbedre dette ved å migrere til en sikrere lagringsløsning, som bruk av HttpOnly og Secure cookies, for å beskytte mot slike angrep og ytterligere styrke sikkerheten i systemet.

// MANGLER VEL NOEN CRUD OPERASJONER?+???????

**CRUD-operasjoner:** Vi implementerte fullstendige CRUD (Create, Read, Update, Delete) operasjoner for anmeldelser, som tillot brukere å effektivt opprette, lese, oppdatere, og slette sine egne anmeldelser på nettsiden. Dette ble understøttet av et godt strukturert backend API og en effektiv databasehåndtering.

**Responsiv og sikker brukerinteraksjon:** Frontend-utviklingen sikret at applikasjonen var tilgjengelig og optimalisert for ulike enheter, og backend-arkitekturen sørget for sikker og - effektiv datahåndtering. Integreringen av frontend med backend via fetch-metoden forbedret brukeropplevelsen ved å muliggjøre dynamiske oppdateringer uten å laste siden på nytt.

#### Feilhåndtering og datavalidering: Vårt system for feilhåndtering og datavalidering på både frontend og backend har sikret at brukerdata behandles korrekt og sikkert. Dette har redusert forekomsten av feil og sikret at dataene som brukes og vises er nøyaktige og oppdateres korrekt over plattformen.

**Brukertest og evaluering:** Gjennom kontinuerlige brukertester (selvtestet) og iterasjoner, har vi forbedret funksjonaliteten basert på identifiserte behov og tilbakemeldinger. Dette har ført til en stadig forbedring av både brukergrensesnitt og funksjonalitet.

**DISKUSJON**

blablabla

blabla

**KONKLUSJON**

blablabla

bla

Mangler:

- "likes"

- muligheten til å følge en annen bruker

- muligheten til å se andre brukere sin profil og deres bidrag

- legge til forbedret sikkerhet rundt registrering ved email confirmation

- BLOB og bildeopplastninger

- Det ble mye surr med forntend og design

- si noe om fremmednøkler og relasjoner til tabellene

- START I INDEX.HTML I ROOT

- vi har lært å bruke GIT og Github