

# IT2810-Webutvikling

GRUPPE 42

Maria Soleim  
Tollef Jørgensen  
Erlend Skarpnes  
Håvard Løkensgard

November 22, 2017

# 1 Applikasjonen

**Discover HS** er en webapplikasjon utviklet med *Angular 4*. Hensikten er å gi en enkel, intuitiv og minimalistisk opplevelse som gjør det enkelt for Hearthstone-interesserte å finne informasjon om Hearthstone-kort.

Applikasjonen vil bli bygd opp av flere forskjellige komponenter med tanke på best praticc ved *Angular4*. Se appendix for flowchart over hovedkomponentene. Dette oppfyller kravspesifikasjon 1. (*Appendix: Kravspesifikasjoner, punkt 1.*)

## 2 Data

Dataen vi skal bruke henter vi fra en egen database. Dataen vil bli hentet ut som JSON-objekter som inneholder informasjon om kortene. Utgangspunktet for dataen er Hearthstone sin egen database. Vi henter ut informasjonen om kortene ved å bruke API'en til Hearthstone.

## 3 Implementert funksjonalitet

- Mulighet for å bytte mellom faner *DiscoverHs*, *Cards*, *Profile*, *Admin* og *Log in/out*
- Mulighet for å sortere kort ved bruk av ikoner på elementene:
  - Class
  - Type
  - Rarity
- Mulighet for å sortere kort med sliders:
  - Attack
  - Health
  - Mana
- Bytte mellom to forskjellige view ved bruk av en toggle knapp *CardView*:
  - *ListView*: Laster inn de seks første kortene ved navn. Flere kort lastes ved å bruke next/previous-knappene. Kortene kan klikkes på for å få opp mer informasjon. Se figur 5.
  - *CardView*: Laster inn bilder av kortene, og beholder funksjonalitetene til *ListView*. Se figur 6.
- Lage en egen bruker
- Se andre registrerte brukere
- Logge inn og ut

### **3.1 Implementasjon som var planlagt, men ble valgt bort under prosjektet**

- Mulighet for å søke etter kort ved navn
- Legge til et favorittkort

## 4 Løsning

### 4.1 GUI

Vi har valgt å ta i bruk biblioteket *bootstrap 4*. Grunnen til dette er at det er enkelt og godt utviklet bibliotek. Under er det lagt ved bilder av designet med forklaring til design valg. Designet har gått i en kontinuerlig iterativ prosess under hele prosjektet.

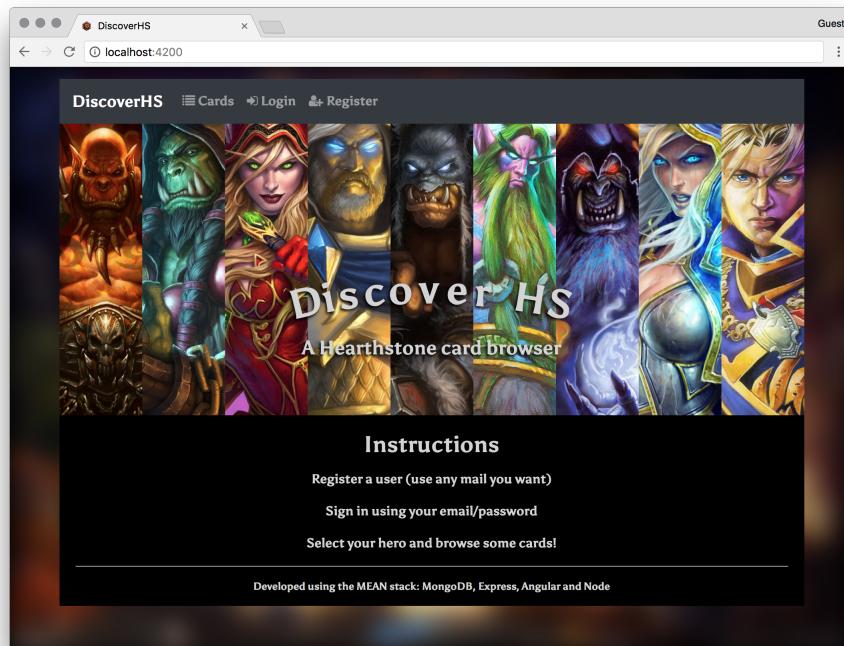


Figure 1: *DiscoverHS* er hovedsiden. Fra denne siden har man mulighet for å navigere seg til de andre sidene. Det er også en noen enkle retningslinjer for hvordan man kommer igang med å bruke siden.

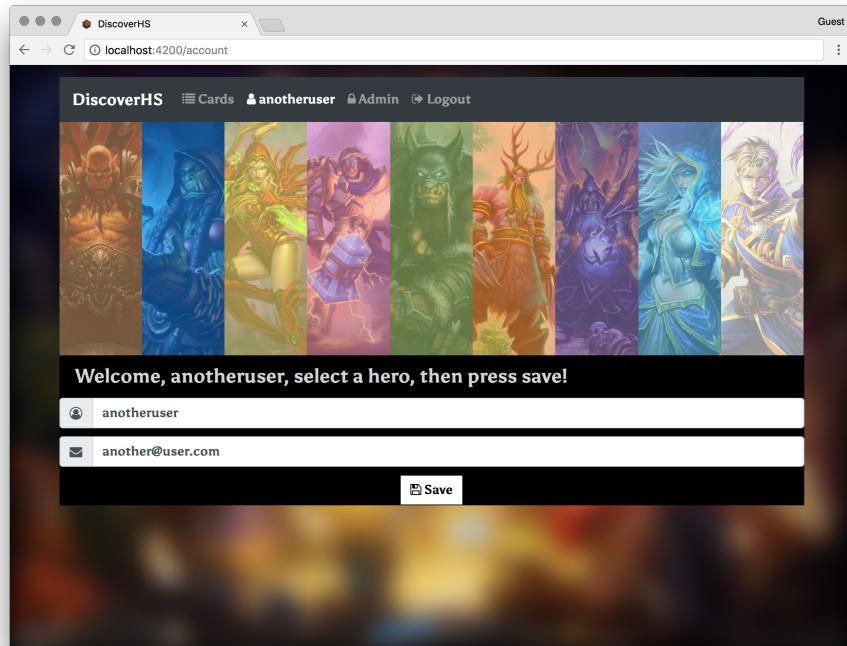


Figure 2: Profile siden. Her er det en oversikt over alle *Heroene* man kan velge. Ved å klikke på en av dem vil den ekspandere og fargene bli klarere. Ved å trykke på save, vil valget bli skrevet til databasen og med det lagret. I tillegg vil fargen på knappen samsvare med fargen til Heroen.

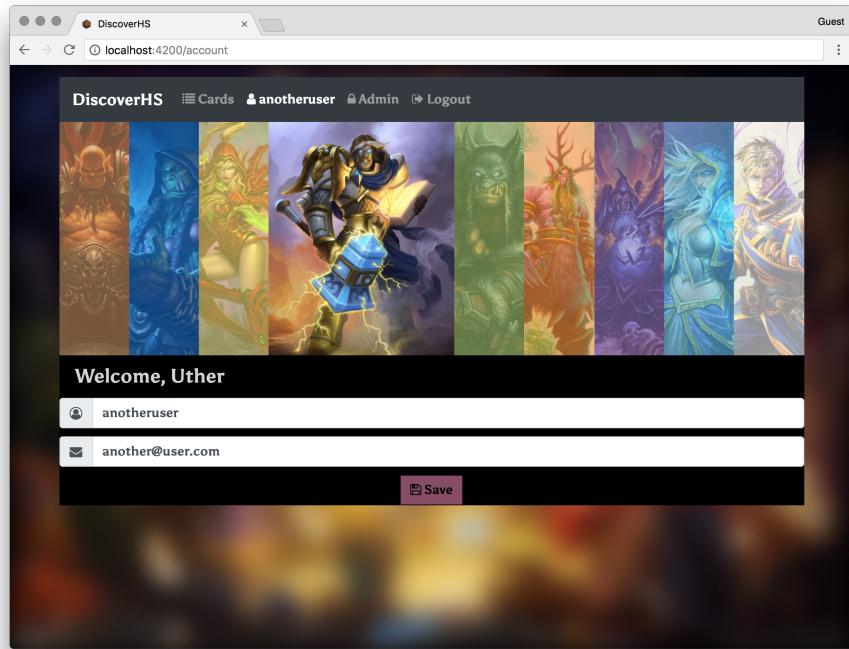
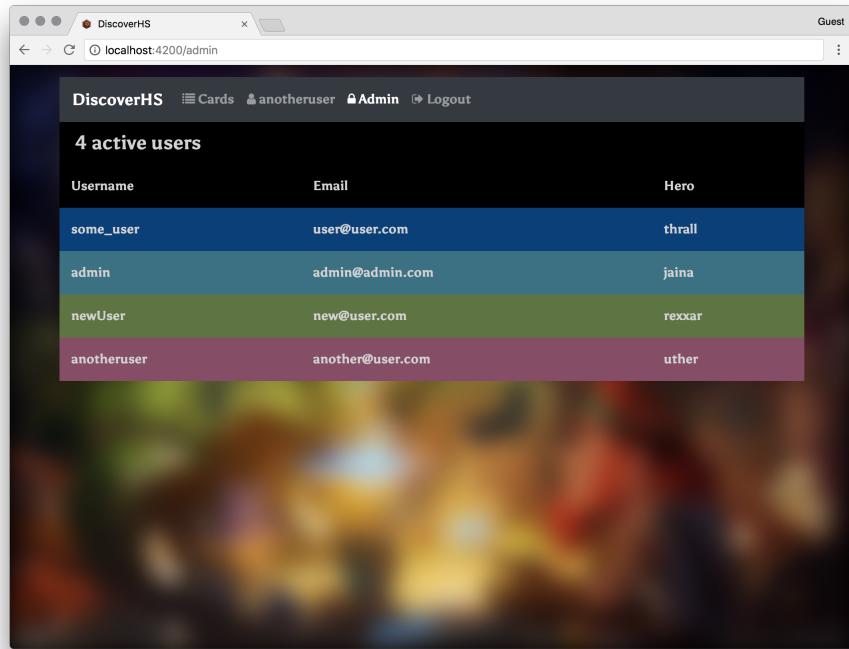


Figure 3: Hero valgt: Uther



A screenshot of a web browser window titled "DiscoverHS" showing a user management interface. The URL in the address bar is "localhost:4200/admin". The page displays a table of "4 active users" with columns: "Username", "Email", and "Hero". The data is as follows:

Username	Email	Hero
some_user	user@user.com	thrall
admin	admin@admin.com	jaina
newUser	new@user.com	rexxar
anotheruser	another@user.com	uther

Figure 4: Liste over alle registrerte brukere med hero-valget deres. I den siste iterasjonen ble email fjernet.

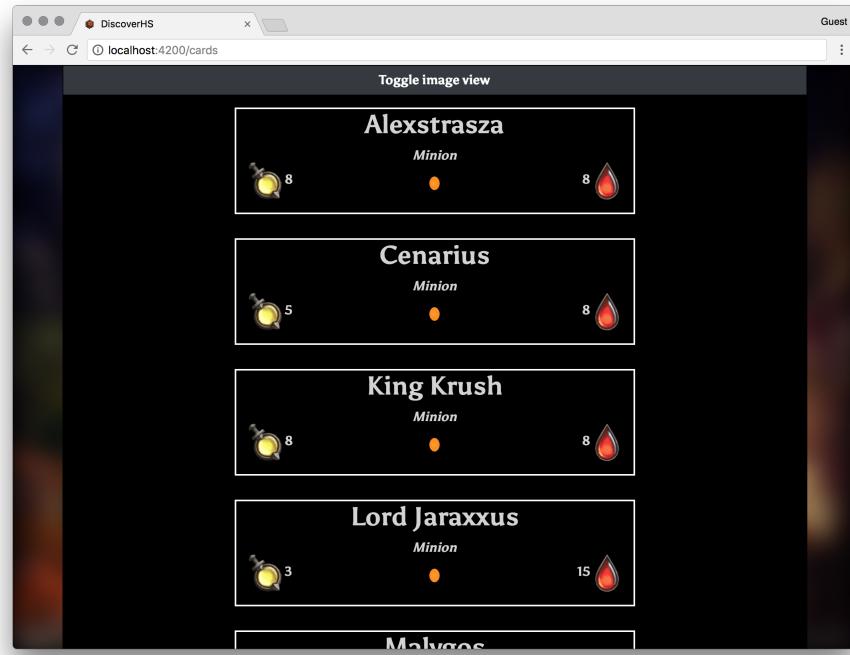


Figure 5: Liste fremvisning av utvalgte kort. Rarity er representert med ikonet i midten, mens attack og hp står på venstre og høyre side.

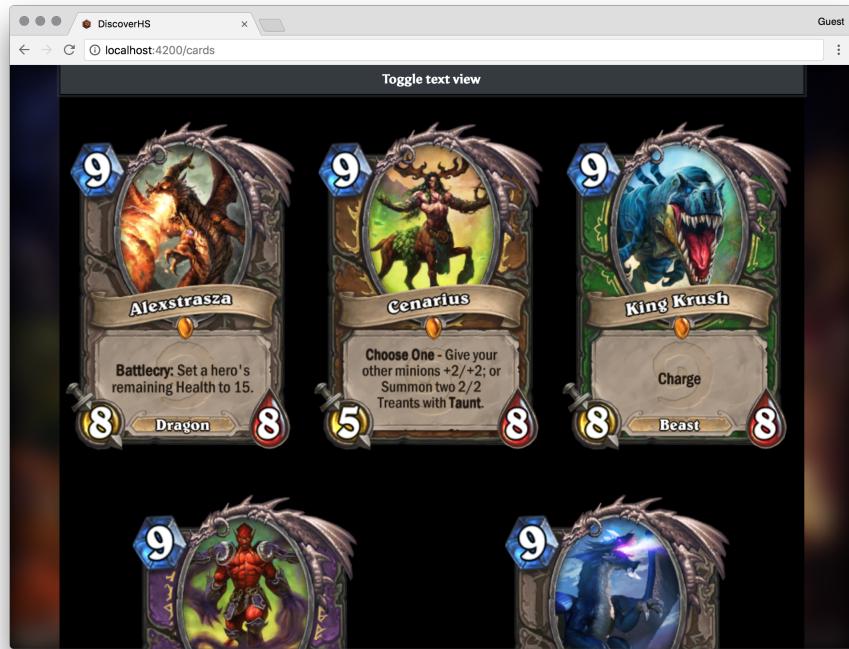


Figure 6: Ved å toggle fra listview til cardview bytter man hvordan kortene blir representert på skjermen.

## 4.2 Database

Vi har valgt å bruke *MongoDB* som løsning for databasen. (*Appendix: Kravspesifikasjon, punkt 2.*) Den inneholder informasjon om alle Hearthstone kort, slik at vi enkelt kan hente ut informasjonen vi vil ha. Bildene til de forskjellige kortene blir hentet fra en annen server, dette er for å unngå at databasen tar for stor plass. Ved å bruke checkboxene og sliderne vil det bli gjort spørninger mot databasen. Ved å velge en hero og trykke save vil valget av hero bli skrevet til databasen. (*Appendix: Kravspesifikasjon, punkt 3.*)

## Appendix

### Flowchart

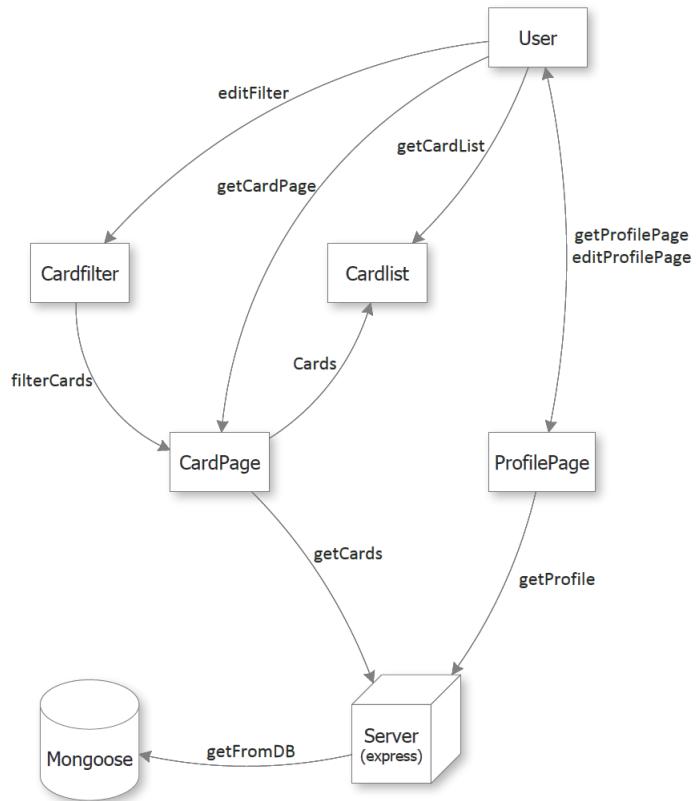


Figure 7: Flowchart over hovedkomponentene

## Kravspesifikasjoner til prosjekt 4

1. Webapplikasjonen skal kjøres på gruppas virtuelle maskin og bruke node.js på serversiden, og skal være utviklet i Angular (bruk v2 eller v4, <https://angular.io>). Det er selvsagt greit å i tillegg bruke andre bibliotek eller løsninger som dere finner hensiktsmessig.
2. I webapplikasjonen skal det inngå en backend database som kjøres på gruppas virtuelle maskin. Type database og hvordan denne brukes er opp til dere å bestemme, men grensesnittet til databasen skal være godt designet ihht. god praksis (bruk av REST ea).
3. Dere skal demonstrere både skriving og lesing til databasen fra webapplikasjonen inklusive en form for søk (i praksis dynamisk brukerdefinert utvalg av det som skal vises). Generelt er det mye artigere å jobbe med en datamengde som gir et realistisk inntrykk (eksempevis mulig å søke på forskjellige ting og få resultatsett som er forskjellige og har forskjellig antall). Bruk data dere finner på web, eller lag egne data.
4. Brukergrensensittet skal ha listebasert visning med få detaljer for hver enhet, og hvor målet er å vise brukeren hva som er i databasen eller hva som er resultatet av et søk. Brukeren skal ha mulighet til å se flere detaljer for hver enhet enten i et eget vindu, eller ved at listen enheten i lista har expand/collapse egenskap.
5. Den listebaserte visningen skal kunne sorteres på minimum to forskjellige egenskaper. Eksempel: etter at brukeren har fått returnert en liste etter et søk skal brukeren kunne bytte mellom forskjellige sorteringer.
6. Den listebaserte visningen skal kunne filtreres på minimum to forskjellige egenskaper. Eksempel: etter at brukeren har fått returnert en liste etter et søk skal brukeren kunne krysse av på en egenskap for å få begrenset antallet enheter i resultatsettet til kun de som har denne egenskapen.
7. Den listebaserte visningen skal ha dynamisk lasting av data. Eksempel: etter et søk vises de 10 første treffene, men flere lastes når brukeren scroller eller ved blaing i sider.
8. Webapplikasjonen skal ha ”min side” funksjonalitet som i praksis betyr at en bruker skal kunne logge seg på og at det blir registrert noe fra brukerens søkeaktiviteten f.eks. hva brukeren har sett på tidligere eller søkerne som brukeren har brukt.
9. Webapplisjonen må implementere ”session”-håndtering (som du f.eks. trenger for å implementere dynamisk lasting, min side, og filtrering/sortering som skal fungere med sidevisning).
10. Webapplikasjonen skal ha et litt ”fancy” alternativ visning av listen f.eks. visning på kart eller visuell grafisk fremstilling av data, ordsky ea.

11. Kode skal være testet og funksjonaliteten skal være godt utprøvd og feilfri.
12. Prosjektet skal være godt dokumentert, slik at det er lett å sette seg inn i for andre.