ITPE3200-1 22H Webapplikasjoner

Project report

Dr.Health

DiagnoseApp

Mohammad Khir Khaled Almohammad s343988
Faesal al shhadat s 339410
Ali Fariz Alghadban s362111

ITPE3200-1 22H Webapplikasjoner

1. Introduksjon / Bakgrunn

Dette er litt om hva prosjektet går ut på og hva dets mål er. Prosjektet er en løsning på problemet med "Symptom til diagnose kalkulatur". Prosjektet er en del av to gruppe innlevenger. Dette skal være en løsning hvor man kan velge forhåndsdefinerte symptomer og utifra det, skal løsningen fortelle eller vise resultater om hva du kan ha(dvs. Din diagnose).
Etter diskusjon og drøfting av problemet har vi kommet på en løsning som kan oppfylle stort sett kravene til prosjektet, som ble oppgitt i prosjektsmal. Vi skal lage en nettside som inneholder prosedyrer og funksjonalitet som tar for seg alle kravene og funksjonaliteten som trengs.

2. Mål

Målet med dette prosjektet er å lage en effektiv webløsning som gjør det mulig for en person/ pasient å kunne diagnostisere seg selv (vite om han/hun har sykdom eller ikke). Dette kan hjelpe brukeren til å forutsi om han bør kontakte legen eller ta grep om å kontakte spesial institusjon for å bekrefte/avkrefte sykdom. I tillegg skal denne webløsning være enkel å brukes.

3. Beskrivelse av prosjektet

Dette er kort beskrivelse om prosjekts funksjonlitet og hva prosjektet den skal kunne promotere. Beskrivelsen tar med stepvis funksjonaliteten av systemet.

Til å begynne med har vi en påloggingsmodul som lar de aktuelle personene (i dette tilfelle person som har registert seg i nettsiden) logge på. Alle bør ha sitt eget dashbord etter innlogging. Innloggingen skal skje ved at personen/pasienten logger inn med eget bruker. De får laget brukeren ved å følge instruksjonen som blir oppgitt. Dette kan spesifiseres i kravspesikasjoner. Denne modulen har vi tenkt å ta med i oppgave2. derfor skal personen/ pasienten kunne bruke nettsiden og dets funksjonalitet uten å kunne ha behov for innlogging.

Etter at personen/pasient har kommet seg til hovedsiden, dukker opp enkelt Home Page hvor det står lit informasjon om sykdommen, som personen skal teste. Først må fylle ut informasjon om seg selv. Dette gjøres ved å fylle ut de nødvendige innputt feltene. Dette skal inneholde (navn, fornavn, adresse, telefonnummer, personnummer og e-post). Disse informasjonene skal kunne lagres i databasen. Nett siden støtter CRUD som gir personen muligheten til å kunne lage, lese, oppdatere og slette disse informasjonen om deg selv.

ITPE3200-1 22H Webapplikasjoner

Når personen har lagret og bekreftet opplysningene, dukker opp alle opplysningene i et nytt window i en tabell som inneholder endre og slett knapper. Personen kan bruke disse knappene for eventuelt å kunne endre eller slette sine opplysninger. Hvis personen vil gå videre for å kunne teste seg, trykker personen <diagnoser> knappen for å navigere til et nytt window som inneholder mange forskjellige symptomer. Personen bør velge de relevante symptomer som beskriver hans situasjon. Etter å ha valgt de relevante symptomene, vil nettsiden kunne vise resultater som viser hva personen kan ha ut ifra disse symptomene i resultat side. I tillegg skal personen kunne se historikken hvis han har tidligere testet seg i denne nettsiden.
NB: Vi tar i betrakting at det er kun en person som bruker nettsiden og derfor vises alle resultatene uten å bruke personId.

Eksempel på noen brukerhistorier så lenge:

personer:

1. personen vil ha mulighet til å åpne nettsiden for å kunne registere seg.
2. personen vi kunne sjekke opplysninene for å kunne se sine info før lagring.
3. personen vil kunne registere seg for å kunne lagre sine informasjon i nettsiden.
4. personen vil kunne lese og oppdatere for å kunne holde sine info oppdatert.
5. personen vil kunne slette sine opplysningere for å ikke ha noen opplysninger om seg selv i sysstemt.
6. personen skal kunne bruke sykdom kalkalturen for å kunne teste seg.
7. personen vil kunne se sine testede resulater
8. personen skal kune se resultat historikk.

Primær flyt:

Tatt i betraktning at brukere er logget inn:

a. persoenen fyller alle innputt feltene.

b. persoenen bekrefter opplysningene.

c. persoenen velger relevante symptomene.

d. persoenen vil se relevante resultater.

ITPE3200-1 22H Webapplikasjoner

Alternativ flyt b:

a.hvis personen velger å slette opplysninger, må han tilbake til punkt a.

b. Hvis personen gir unøyaktig informasjon, må han gå tilbake til punkt a. (dette punktet tas med senere når det blir implimentert validering for innputtfeltene)

Alternativ flyt d:

b. personen avviser resultater, må personen tilbake til punkt c.

4. Utvikling av krav.

For å kunne identifisere prosjekt krav måtte alle teamsmedelemene utførske og drøfte nødvendlige og relevante krav som kan oppfylle hovedkravene i oppgave teksten. I tillegg la vi til noen andre ideer som kan komme i bruk og blir implimetert i neste del av oppgaven. I den føste delen tenkte vi å srtarte med kun en type sykdom, men til neste oppgave, vil vi utvide det til et større prosjekt med med andre sykdommer og ekstra funksjunalitet.

5. Kreavspesifikasjoner.

Vi brukte KISS-metoden for å holde bedre oversikt over kravspesifikasjonene og som gir mer ryddigere adminesrtativ struktur for prosjektet.

K-I-S-S	Bruker Stories	Løsning	Prioritet
Improve	**Jeg vil kunne registere meg som bruker.** Bør være et alternativ selvom vi ikke skal ta med logginn siden i føste prosjekt.	Det bør være en løsning hvor at brukerern kan fylle ut et skjema med personlige opplysninger	Høy
improve	**Jeg vil kunne se mine opplysninger.** Det skal være mulighet for å sjekke opplysninger før lagring.	Det bør være en løsning hvor at brukerern kan trykke en knapp for å sjekke de registerte opplysningene.	Høy

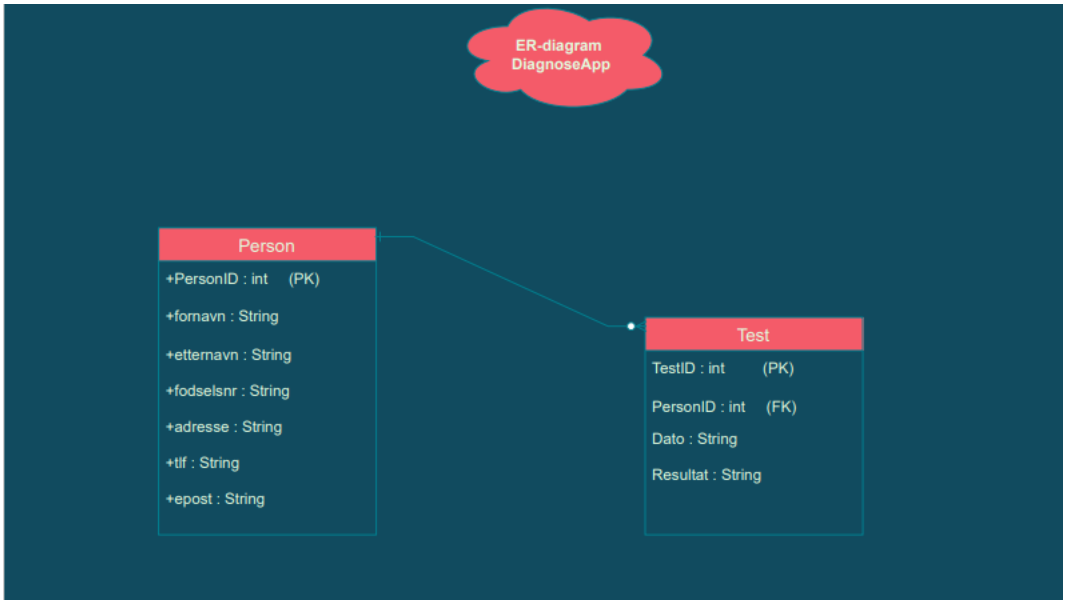
ITPE3200-1 22H Webapplikasjoner

Improve	**Jeg vi ha mulighet for å kunne endre og slette mine opplysninger.** det burde gjøres under oversiktten over personlige opplysningene.	Det bør være en løsning hvor at brukerern kan trykke en knapp for å kunne endre de registerte opplysningene, og en knapp for å slette opplysningne	Høy
Improve	**Jeg vil kunne se oversikt over relevante symptomer.** Bør komme i egen side.	Det skal implimenteres flere forksjellige typer av symptomer relatert til den vagte sykdommen.	Høy
improve	**Jeg vi kunne bruke diagnose kalkulatorern og kunne velge fra de utgitte symptomene.** Bør kunne være enkel å bruke	Personen skal kunne velge symptomer ved hjelp av radio-knapper og sjekkbokser.	Høy
Improve	**Jeg vil kunne se mine resultater fra diagnose kalkulatoren.** Bør komme i egen side.	Det skal være mulig og se resultatet samt med forklarenede tekst om sykdommen.	Høy
Improve	**Jeg vil kunne se mine tidligere tester.** Bør kunne se alle resultater	Det skal være mlig å se tidligere resulatater samt med person info i result side.	Medium

ITPE3200-1 22H Webapplikasjoner

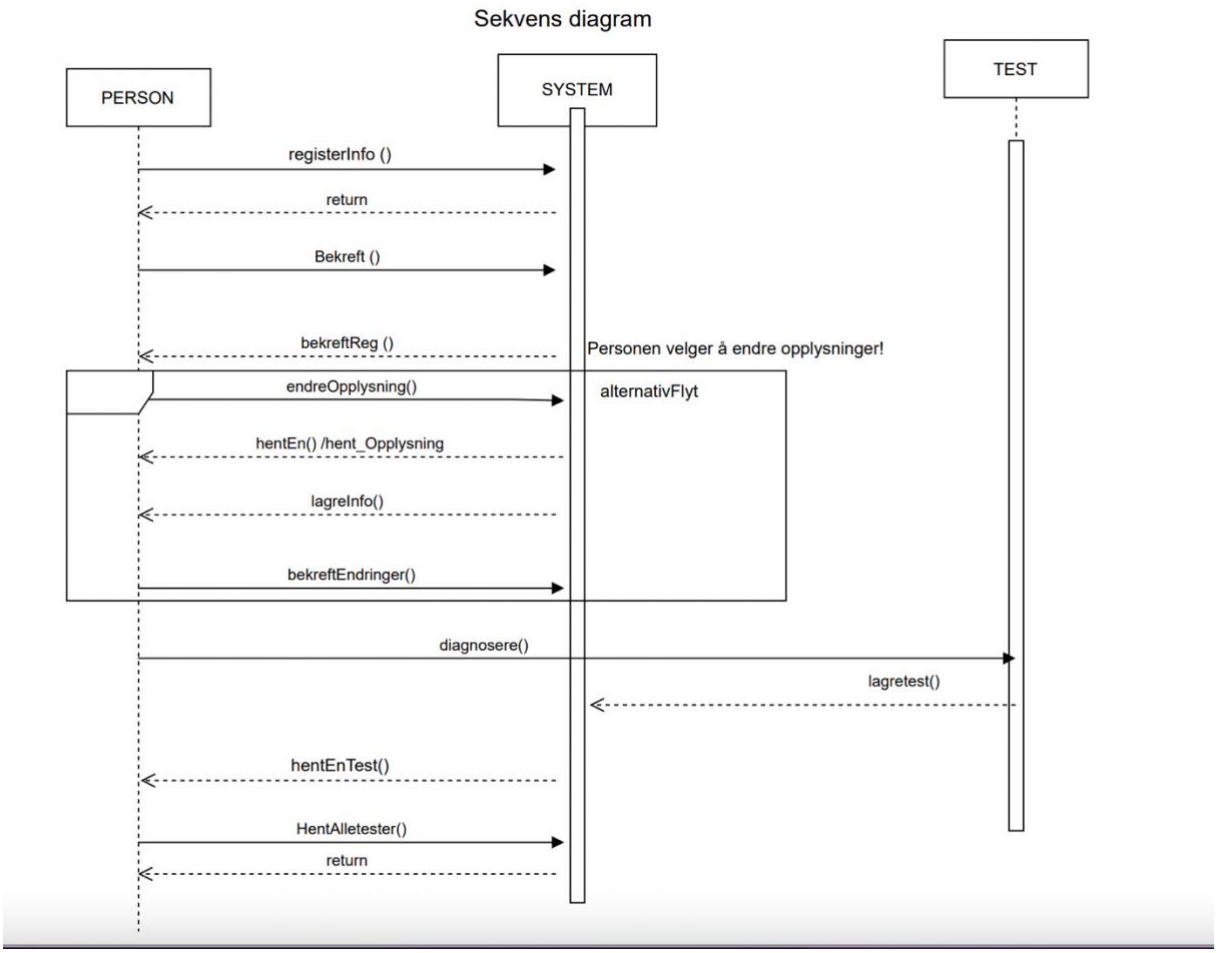
6. UML-diagramer :

ER_diaram:

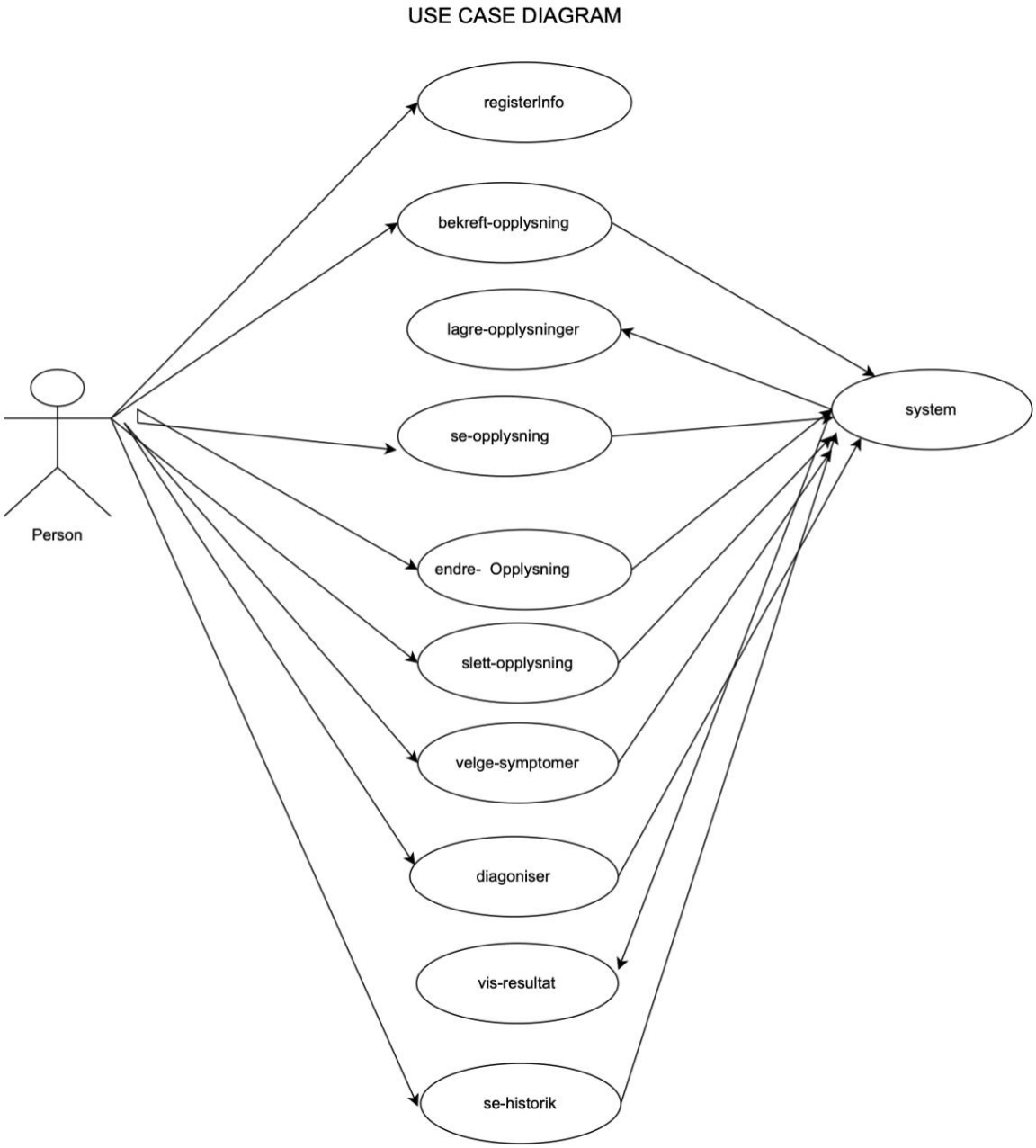


ITPE3200-1 22H Webapplikasjoner

Sekvens diagram:



Use case diagram:



7. Utfordringer:

PLANLEGGING og GJENNOMFØRIMNG

- Det kan være litt utfordrende å holde motivasjonen på samme tempo gjennom helle prosjekts gjennomføringsfasen. Ved starten har man veldig god optimisme spesielt ved planlegging , men etter hvert når oppgavene settes i utførings fase, kan det enten bli lite eller for mye å gjøre. I tillegg kan det også noen ganger bli mye annet å gjøre på samme tid i vanlig livet som for eksempel: jobb ved siden studien, andre

fag å oppfølge i tillegg til prosjktet eller andre saker i livet rundt. Dette kan dermed skape litt utfordinger med tanke på levering eller utføring av oppgavene i prosjektet og videreføre til litt forsinkelser.

CODING:

- Lage relasjoner mellom tabellene i databasen. Vi trengte å koble Person.tabell med Test.Tabell for at resultaten skulle være knyttet til person info. Vi brukte lang tid på kunne finne en løsning for dette problemet. Dette trengte vi for å kunne vise resultatet til den personen som tester seg samt med personinfo på samme side. Vi gjørde det på en møte ved å sette FK(personid) i tabellen Test.Vi brukte en modellklass (Result) mellom klient og server som inneholde alle attributtene. Dette brukes også for å vise i resultat tabell når resultatene kalles.

  For å nå til personid og testid sammen måtte vi bruke denne koden som vi har fått inspirasjonen fra nettet:

  https://www.codexpedia.com/javascript/javascript-get-the-query-string-param-from-url/

```
var params = {},
    uriparams = location.search.substr(1).split('&');
for (var i = 0; i < uriparams.length; i++) {
    var parameter = uriparams[i].split('=');
    params[parameter[0]] = parameter[1];
}
```

SAMMARBEID:

- Gruppen er dannet av tre studenter fra forskjellige årsstudien 2/3. Dette skaptet dermed utfordringer med å kunne finne passende tid for sammarbeid i prosjektet.

- Siden dette var første gang for gruppen å jobbe med en fullstak prosjekt, førte det til utfordinger med å kunne dele oppgaven på slik møte at alle kunne klare å koble til andres oppgaver. eks (når det gjelder frontend, var det vanskelig for en medlem i gruppen å jobbe i samme oppgave sammtidig når den andre jobber med). For å kunne håndtere denne utfrodringen , måtte vi alltid komentere kodene slik at alle som er med i gruppen forstår hva denne koden gjør, dersom han skal videre utvikle.

TEKNISKE UTFORDRINGER:

- En anenn utfording var at vi brukte forskjellige versjoner av target fremwork. .NET 6.0 og .NET Core 3.1. For å unngå forventede konflikter, bestemte vi å bruke .net core 3.1.

- Siden vi brukte Github måtte vi lage en repostory slik at alle får tilgang til den. Utfordringen var at det kjedde ofte konflikter gjennoem pushing og pull. Probleme er at når en av gruppen prøvde å Clone/ Pull til disktop, får ikke alle endringen som ble pushet til github. Vi måtte bruke lang tid på å fikse slike problemer.

8. Prossess:

Vi brukte en lokal database (SQLite). Dette er enklere å bruke i C# som opprettes når programet kjøres første gang. Det må legges til en link i Startup filen for å koble med SQLite. Vi brukte ORM for å akssessere database. Object Relational Mapper gjør det enklere å skrive kode mot database enn å akssessere database med vanlige SQL-spørringer.

Vi brukte asynkrone for programmering mot database. Asynkrone brukes for å håndtere store mengder av trafikk inn i systemet mot database. Vi kjørte asynkron selvom vi ikke har store megde trafikk i Applikasjoonen vår, men på grunn av at dette har blitt standard å bruke da brukte vi asynkrone her uansett.

9. Kildelist

- Vår hoved kildet er pensommet som ligger på canvas.
- https://www.w3schools.com/cs/index.php

ITPE3200-1 22H Webapplikasjoner

- https://www.codexpedia.com/javascript/javascript-get-the-query-string-param-from-url/
- https://learn.microsoft.com/en-us/ef/