Høgskulen på Vestlandet

Fakultet for ingeniør- og naturvitskap

Institutt for datateknologi, elektroteknologi og realfag

Administrasjonsverktøy til kontroll og videreutvikling av chatbot

Systemdokumentasjon

Versjon <1.0>

Dokumentet er basert på Systemdokumentasjon utarbeidet ved NTNU. Revisjon og tilpasninger til bruk ved IDER, DATA-INF utført av Carsten Gunnar Helgesen, Svein-Ivar Lillehaug og Per Christian Engdal. Dokumentet finnes også i engelsk utgave.

REVISJONSHISTORIE

Dato	Versjon	Beskrivelse	Forfatter
25/04/23	1.0		Ole Anders

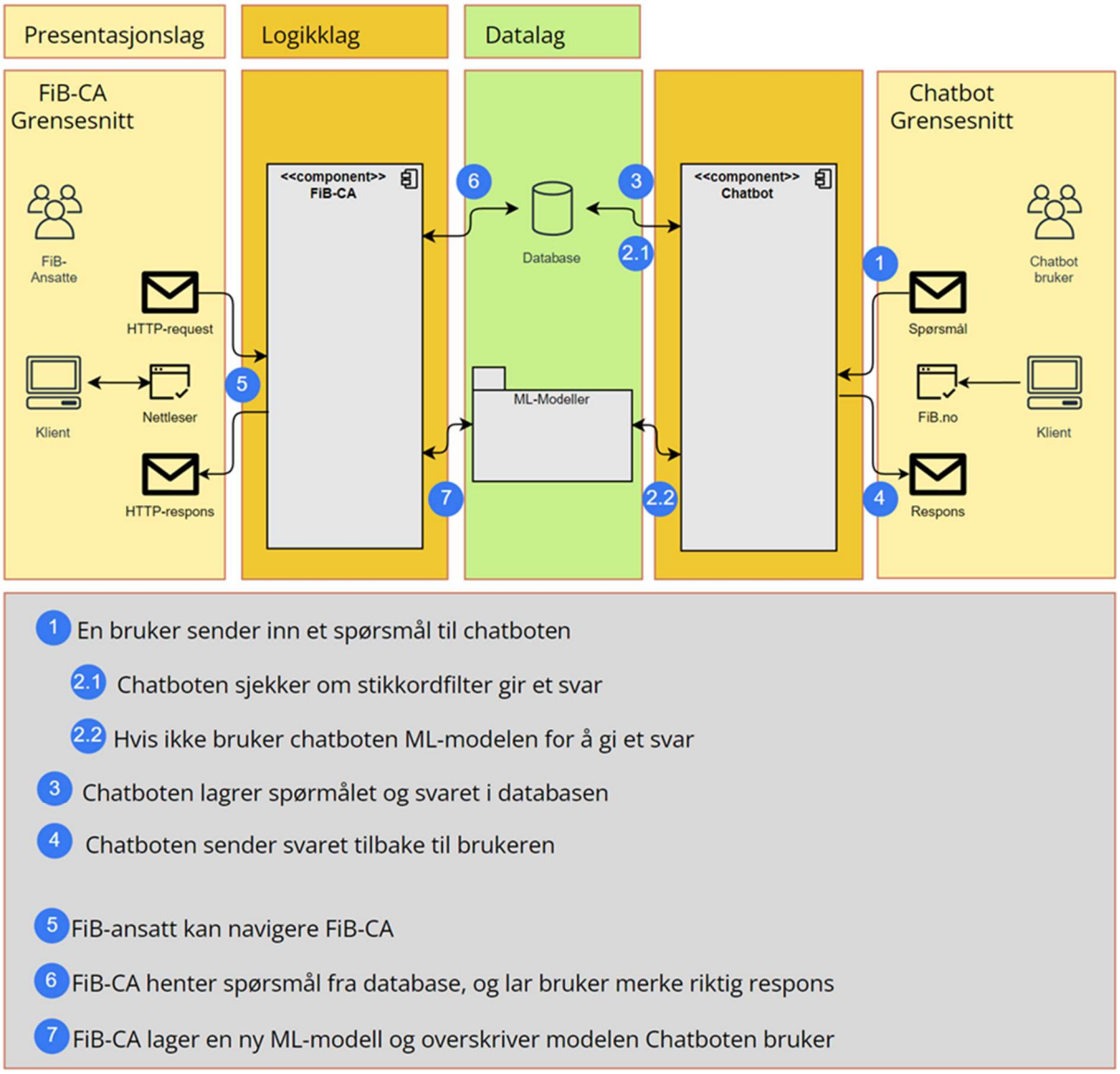
INNHOLDSFORTEGNELSE

1 INNLEDNING

Dette dokumentet inneholder systemdokumentasjon av FiB-CA. Hensikten med dokumentet er å beskrive hvordan FiB-CA er designet og laget, og gi nødvendig innsikt i programmet for å kunne vidreutvikle og vedlikeholde det.

2 Arkitektur

Løsningen for dette prosjektet er bygget på en tre-lags arkitektur. Den deler datalag med den allerede eksisterende chatboten til FiB. Begge programmene er bygget i et Flask rammeverk

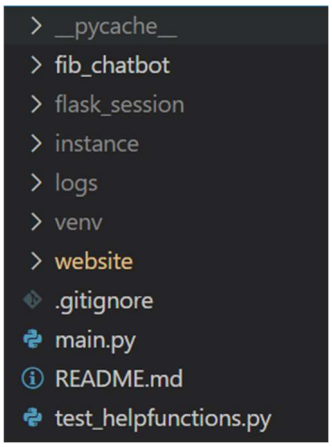


Denne figuren beskriver hendelser der de forskjellige lagene kommuniserer med hverandre

For øyeblikket er datalaget for FiB-CA lagret internt i programmet, og er ikke koblet opp mot FiB sin chatbot.

3 PROSJEKTSTRUKTUR

Den overordnede strukturen til prosjektet vårt er som definert under

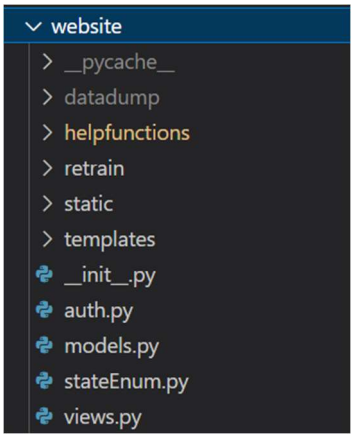


3.1 fib_chatbot

Fib_chatbot mappen inneholder en versjon av koden for FiB sin chatbot med alle de nødvendige endringene for at FiB-CA skal fungere sammen med den.

3.2 website

Website mappen inneholder pakken for FiB-CA. __init__.py er filen som definerer website mappen som en python pakke og gjør at denne kan initieres.



3.2.1 auth.py

Auth.py er filen som beskriver ruter assosiert med autorisering av en bruker. Her blir handlingen for alle http-forespørslene som omhandler dette definert.

3.2.2 models.py

inneholder alle klassene som beskriver en tabell i databasen. Disse objektene blir brukt for å lett kunne håndtre informasjon fra databasen

3.2.3 stateEnum.py

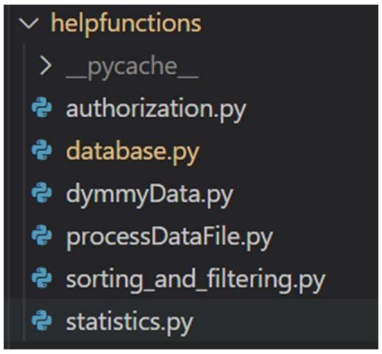
inneholder alle enumene som blir brukt i programmet.

3.2.4 views.py

Views.py inneholder rutene for hver av sidene i FiB-CA som ikke omhandler autorisasjon av brukere. Her blir handlingen for disse http-forespørslene definert

3.2.5 helpfunctions

3.2.5.1 authorization.py



Hjelpemetodene fra authorization.py er lagd for å sperre tilgang for brukere som ikke er innlogget, og logge ut inaktive brukere

3.2.5.2 database.py

All kommunikasjon med databasen skjer gjennom hjelpemetoder herfra. Database.py er et sett med metoder tilpasset vårt innhold i databasen

3.2.5.3 dummyData.py

Alle metodene i dummydata er laget for å generere data for databasen. Denne klassen vil ikke være nødvendig i den endelige løsningen, men er grei å ha for å feilsøke programmets evne til å håndtre forskjellig data

3.2.5.4 prosessDataFiles.py

Metodene i prosessDataFiles.py inneholder metoder som tar hånd om at preprosessering og trening på data forekommer riktig

3.2.5.5 sorting_and_filtring.py

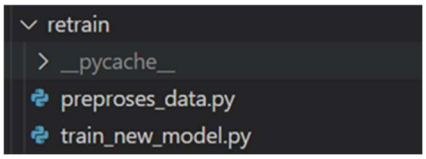
Metodene i sorting_and_filtring.py er alle metodene brukt for enten sortering eller filtrering av lister med forskjellige diverse datatyper.

3.2.5.6 statistics.py

Denne klassen inneholder diverse metoder for å beregne statistikk og fremvist data

3.2.6 retrain

3.2.6.1 preprosess_data.py



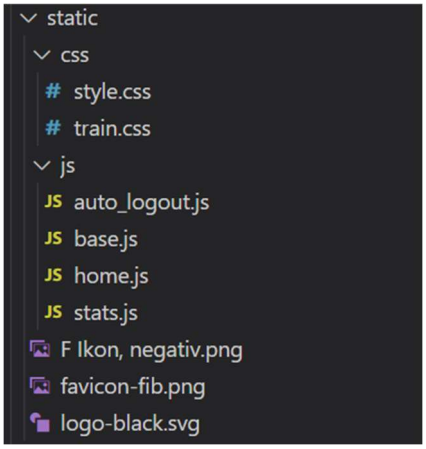
inneholder alt nødvendig for å klargjøre en liste med spørsmål som treningsdata og lagre dette i en csv fil

3.2.6.2 train_new_model.py

inneholder alt nødvendig for å trene en vectoriser og en blender modell med hjelp av en csv fil med reprosessert data

3.2.7 static

3.2.7.1 css



CSS mappen inneholder alle CSS-filene til prosjektet utenom Bootstrap sin. Bootstrap sin CSS blir referert til med en url link i base.html

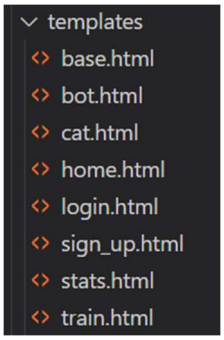
3.2.7.2 js

Det er blitt brukt JavaScript for å fremvise deler av informasjonen på enkelte sider. Alle JavaScript filene er lagret i denne mappen

3.2.7.3 Images

Alle bilder som fremvises i FiB-CA er lagret i static mappen

3.2.8 template



Template mappen inneholder alle html-filene. Dette er alle filene som definerer innholdet på siden.

3.2.8.1 base.html

Base.html er en html-fil som alle de andre filene bygger ut fra. Denne filen inneholder ting som er felles for alle sidene. Dette innebærer blant annet navigasjonsbaren, headeren til html-sidene, og nødvendige CSS og JavaScript referanser.

3.2.8.2 bot.html

bot.html beskriver en side som er lagd for å teste påvirkningen FiB-CA ville hatt på en faktisk chatbot. Dette er ikke en side som trengs å inkluderes i FiB-CA, men som gjør feilsøking lettere.

3.2.8.3 cat.html

Cat.html beskriver siden for å administrere svaralternativer i FiB-CA.

3.2.8.4 home.html

Home.html beskriver hjem-siden i FiB-CA. Dette er siden der brukeren kan gå igjennom hvert enkeltspørsmål og verifisere disse.

3.2.8.5 login.html

Login.html beskriver siden for å logge inn en bruker. Dette er viktig for at

3.2.8.6 signup.html

Signup.html beskriver siden for å opprette en ny bruker i FiB-CA

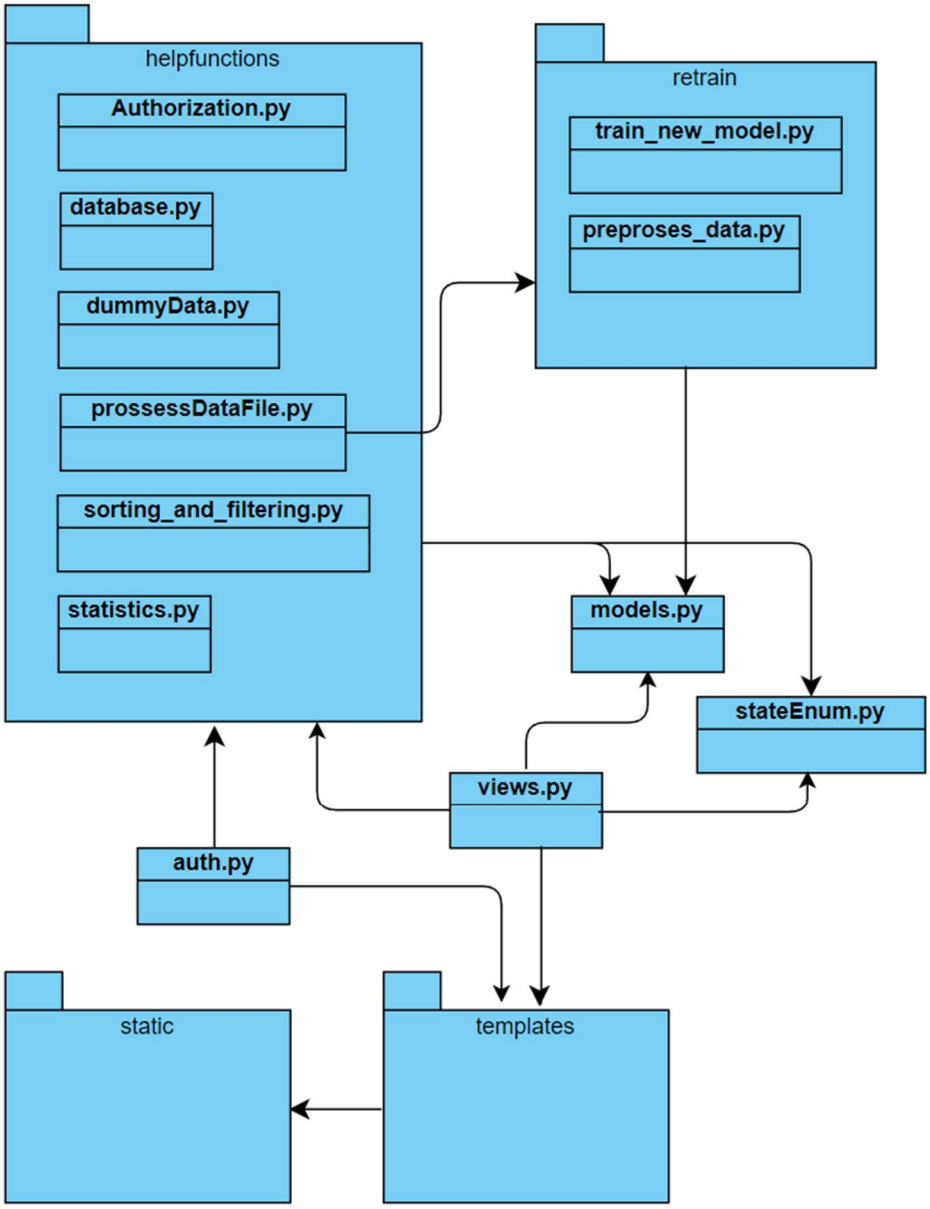
3.2.8.7 stats.html

Stats.html beskriver statistikk sidene.

3.2.8.8 train.html

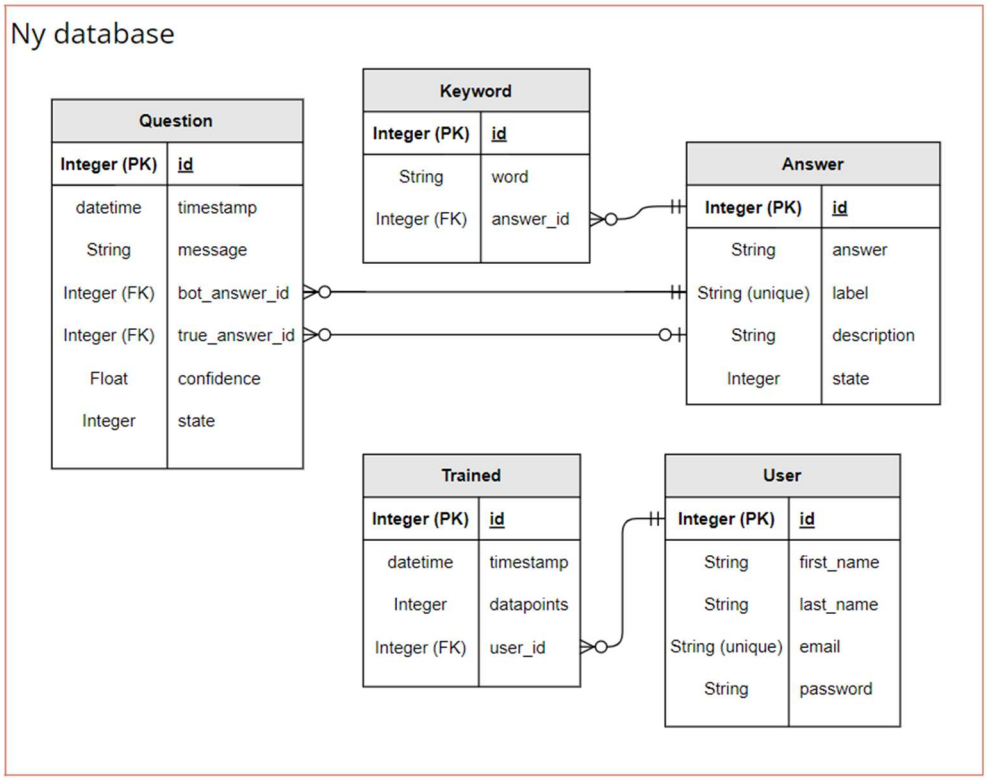
Train.html beskriver tren siden til FiB-CA. Dette er siden der en bruker kan trykke på en knapp for å trene en ny maskinlæringsmodell.

4 KLASSEDIAGRAM



5 DATABASEMODELL

Det er kun en database som er blitt brukt for dette prosjektet



5.1 Question

I question tabellen for databasen blir alle spørsmål chatboten mottar lagret. Det eneste FiB-CA endrer på av verdier i denne tabellen etter chatboten har lagret et spørsmål der er true_answer_id og state. Dette er for at FiB-CA skal kunne merke hva som er riktig svar på spørsmålet og huke av at spørsmålet er blitt validert eller trent på når dette skjer.

5.2 Answer

Answer tilsvarer et svaralternativ til chatboten. Alle svaralternativene til chatboten blir lagret i denne tabellen uavhengig av hvilken type svaralternativ det er. Om et svaralternativ er aktivt eller inaktivt og om det er et sesongsvar eller standardsvar blir beskrevet av state verdien.

5.3 Keyword

Et sesongsvar kan ha et ubestemt antall nøkkelord. Disse nøkkelordene blir brukt i en filtreringsfunksjon, og gjør at chatboten gir svaralternativet med et nøkkelord knyttet til seg om spørsmålet også inneholder dette nøkkelordet

5.4 User

I User tabellen blir alle brukerene til FiB-CA lagret.

5.5 Trained

Trained tabellen inneholder informasjon om hver gang chatboten er blitt trent, hvor mange spørsmål den ble trent på og hvem som gjorde dette. Det var opprinnelig tenkt å lagre maskinlæringsmodellen som ble produsert her også, men dette ble ikke gjort ettersom ikke alle databasetyper støtter lagring av filer som disse.

6 Server-tjenester

FiB-CA bruker kun http-forespørsler, og godtar kun GET og POST forespørsler. Alle http forespørsler blir håndtert i views.py eller auth.py. Uavhengig av hvilke type http-forespørsel serveren mottar blir brukeren verifisert med hjelp av biblioteket Flask-Login for å forsikre seg om at brukeren har lov til å nå den siden.

GET-forespørsler blir brukt når det er ønskelig å se innhold på en side uten å motta ny informasjon fra brukeren.

POST-forespørsler blir brukt for alle operasjoner som fører til endringer på serversiden. Alle POST-forespørsler inneholder et attributt; form, som sammen med url-en viser til riktig handling. De andre attributtene til en POST-forespørsel varierer ut fra hvilke handling som skal gjøres.

7 SIKKERHET

Flask-Login er brukt for autorisering av brukere. Dette er er bibliotek som kan brukes for å logge inn, ut og for å ta vare på brukere sin «session». Det blir brukt en «secret key» for å kryptere og dekryptere «sessions» for brukere. Passordene lagret i databasen er kryptert med sha256 kryptering

Løsningen er beskyttet mot SQL-injections ettersom alle spørringer mot serveren blir formulert av innebygde metoder i biblioteket flask-sqlalchemy og ikke som SQL queries.

Ettersom løsningen ikke enda er satte opp på en server er ikke brannmurinstillinger, tilgjengelighet av porter eller rettigheter til forskjellige brukere på serveren relevant enda, og er vanskelig å si noe om uten informasjon på hvilke type server som vil bli tatt i bruk.

8 INSTALLASJON OG KJØRING

For å kunne kjøre programmet på egen maskin må en installere programmer og pakker. Stegene for å kunne kjøre programmet blir vist nedenfor.

1. **Installere:** *Visual Studio Code*

   For å kunne kjøre programmet gjennom en kodeeditor må du laste ned dette om du ikke allerede har en tilgjengelig. Vi anbefaler Visual Studio Code og vil i videre trin ta utgangspunkt i at dette er kodeeditoren du har valgt å bruke. Visual Studio Code kan lastes ned her: https://code.visualstudio.com/download. Pass på at du velger versjonen for ditt operativsystem

2. **Installere:** Python 3.10

   FiB-CA er utviklet i Python og krever at du har dette på maskinen du skal kjøre programmet på. Det er mange måter å laste ned Python på din maskin, men hvis du ikke allerede er kjent med disse vil vi anbefale du laster det ned gjennom nettsiden deres. Her er en link for å lasten ned Python 3.10: https://www.python.org/downloads/release/python-3100/. Pass på at du velger versjonen for ditt operativsystem

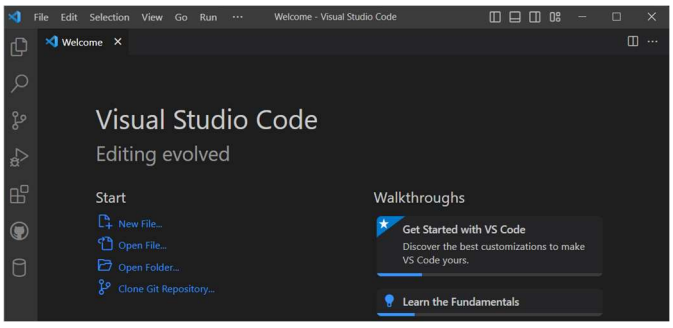
3. **Installere:** Git

   Git er et verktøy for versjonskontroll. FiB-CA er utviklet i et git-repository og vil derfor kreve git for å kunne importere repositoriet. Git kan lastes ned gjennom denne linken: https://git-scm.com/downloads. Pass på at du velger versjonen for ditt operativsystem. Om git aldri er blitt satte opp på pcen din før må dette også gjøres.

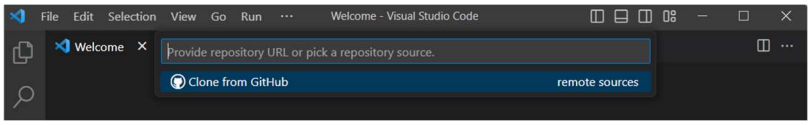
4. **Importer kode fra:** *GitHub*:

   Koden for prosjektet kan hentes fra GitHub. Dette krever at du har tilgang til GitHub-repositoriet. Om du ikke har tilgang til dette kan du forhøre deg med Marek Vetter fra Bouvet, så vil han ta en avgjørelse på om det er grunnlag for å gi deg dette. Om du har tilgang til repositoriet kan du gjøre som beskrevet under for å hente prosjektet

   Første gang du går inn på vscode vil sansynligvis vinduet ligne det under



   Under start, trykk på «Clone Git Repository…» eller trykk deg inn på «Source Control», det tredje ikonet i menyen langs venstre siden av vinduet, og trykk på «Clone Repository» derfra. Det vil poppe opp et lignende alternativ for deg.



Velg «Clone from GitHub» og lim inn følgende link for å fortsette. https://github.com/MatiasBouvet/FiB-CA.git. Om dette er første gang du henter et repository fra GitHub vil VS Code be deg om å logge inn på GitHub for å fortsette. Følg trinnene for å gjøre dette. Etter å ha gjort dette vil du ha repositoriet tilgjengelig på din maskin der du valgte å plassere det.

5. **Installere:** pakker:

   Vi anbefaler bruk av et virtuelt miljø på maskinen din slik at versjonene på pakkene du laster ned ikke påvirker andre program som også bruker disse pakkene på din maskin.

   For å opprette et virtuelt miljø må en først finne adressen til hvor en lastet ned pyton 3.10 og kopiere adressen for så skrive følgende (bytt ut adressen med egen adresse):

   - path\to\Python310\python -m venv venv

   For å aktivere det virtuelle miljøet, skriv i terminal vinduet:

   - venv\Scripts\activate

   Du kan nå installere pakkene en trenger med å skrive følgende i terminalen.

   For å kunne kjøre FiB-CA:

   - pip install -r requirements.txt

   For å kunne kjøre den modifiserte versjonen av Chatboten:

   - pip install -r .\fib_chatbot\requirements.txt

   For å kunne kjøre testklassene:

   - pip install pytest
   - pip install Flask-Testing

6. **Kjøring** av programmet:

   For å starte programmet erstatt «C:\your\path\to\program\main.py» med din destinasjon av plasseringen til main.py i følgende kommando:

   py C:\your\path\to\program\main.py

   eller last ned en python-extention for kjøring av python filer i VS Code og naviger deg til main.py. Da vil du få et «Run»-alternativ øverst i høyre hjørnet. Etter å ha startet programmet kan du gå inn på følgende link i din nettleser: http://127.0.0.1:5000

7. **Legg til data:**

   Ettersom programmet foreløpig ikke er koblet opp mot en reel database må du fylle en lokal versjon av databasen med noe fiktiv data for å se hvordan programmet fungerer. Gå til http://127.0.0.1:5000/bot og bruk nedtrekks-menyen på høyre side

av skjermen for å velge «answers» og trykk på «add to database», også velg «questions» og trykk «add to database» igjen. Da vil du få en liten mengde fiktiv data i programmet ditt.

9 DOKUMENTASJON AV KILDEKODE

Funksjonene for dette programmet er kommentert i kildekoden. Alle funksjoner viser input og output type om de har dette, og har en kort forklaring til seg.

```
def get_keywords_by_answer(answer: Answer) -> List[Keyword]:
    """Return all keyword to a given answer in the form of a list"""
```

Grunnet at kildekoden ikke skal publiseres er det ikke tatt med informasjon om hver av funksjonene her.

10 KONTINUERLIG INTEGRASJON OG TESTING

10.1 Kontinuerlig integrasjon

Det har ikke blitt brukt kontinuerlig integrasjon av produktet opp mot produksjon. Dette kommer av flere grunner:

- Vanskelig å få de rette tillatelsene, grunnet at det er en tidkrevende prosess å få tillatelse, og når en først har fått tillatelse tar det lang tid å få noe til produksjon siden en må være helst sikker på at den nye koden ikke får chatboten til å krasje.
- Alt som skal til produksjon, må være i produksjon minst 1 måned før festspillene begynner
- Den oppdaterte chatbot modellen som prosjektet tar utgangspunkt i ikke er i produksjon
- Det nye systemet trenger nye databasestruktur

Det har derimot blitt lagt vekt på å ha en fungerende versjon på «main», og å bruke andre grener til utviklingen av deler av produktet før endringene blir sendt til «main».

10.2 Testing

Under utviklingen av produktet ble det skrevet tester for å gjøre programmet mer robust, og for at det i ettertid skal være lettere å gjøre endringer i koden. Testene ble skrevet i «flask_testing», som er en utvidelsespakke som gir mulighet for unit testing for flask apper (FlaskTesting, 2017). Det er totalt 23 tester som tester hjelpefunksjonene i «sorting_and_filtering.py» og «statistics.py». Det har derimot ikke blitt laget tester for hjelpefunksjoner i «databse.py», «authorization.py» og «processDataFile.py». Det er heller ikke blitt laget tester for å simulere aktiviteter på selve html-sidene.

10.2.1 Test metoder

Sortering og filterings funksjoner

- test_is_trainedOn()
- test_is_validated()
- test_is_active()
- test_sort_questions_by()
- test_filterQuestionsWithTimestamp()
- test_filterQuestionsWithBotAnswer()
- test_filterByAnswerLable()
- test_is_season()
- test_is_safe()
- test_to_lable()
- test_correct_label()
- test_toggle_answer_state()

Funksjoner til å lage statistikk

- test_getAccuracy()
- test_ConfusionMatrixToDim()

- test_Recall()
- test_precision()
- test_get_f1_score()
- test_get_num_of_questions()
- test_get_the_increase_of_question()
- test_get_number_of_years()
- test_generate_colors()

Andre tester

- test_login()
- test_index()

10.2.2 Kjøring av testene:

- Følg anvisning 1-5 under punkt 8
- Skriv følgende linje I terminalen: py -m unittest test_helpfunctions.py

En normal kjøring av testene vil avslutte med å returnere antall kjørte tester og ordet «OK», er det derimot noen tester som feiler vil testen returnere ordet «Failed» og antall tester som ikke ble godkjent.

11 REFERANSER