# Dag 1

Første dagen besto av fremføring av oppgave, deretter hovedsakelig av planlegging av oppgaven og perioden. Dette ble levert inn før tiden som ble gitt. Deretter brukte jeg litt tid på å sette opp ett [github repo](https://github.com/RasmusLNetland/Fagprove-Rasmus/tree/main), og startet med oppsett av Database. Jeg satt opp en MSSQL container i Docker Desktop, opprettet en bruker (shopp\_app\_user), ett schema (ShopApp), og to roller (AuthManager, DataManger). I dette tilfellet har rollene ingenting å si, ettersom at den eneste brukeren i databasen har tilgang til begge rollene, men hvis vi i fremtiden lager flere brukere så er det ønskelig å kunne filtrere hva brukeren har tilgang til av Stored Procedures. Tabellen ShopApp.User ble også opprettet.

# Dag 2

Dag 2 started ved å fortsette der dag 1 slapp, jeg fortsatt på database infrastrukturen. Tabeller for lister, varer og refreshtokens ble opprettet. Deretter opprettet jeg ett prosjekt for APIet, og startet med setupet der. Jeg la inn diverse pakker jeg visste jeg trengte:

* Mediatr – for å ivareta CQRS pattern, som også separerer koden mer ifra hverandre hvor enkelte deler er ansvarlige for sitt.
* Serilog – Logging verktøy
* SqlClient – Database kommunikasjon
* Polly – For å «retry» database connection.

Jeg satte også opp en lett mappestruktur, som er viktig for å holde orden i prosjektet. Deretter begynte jeg å konfiguerere swagger dokumentasjonen, jeg la inn versioning som er greit om man i fremtiden skal endre på ett endepunkt, men ønsker å beholde den gamle funksjonaliteten. Da kan man opprette en «v2» av samme endepunkt. Deretter startet jeg på Login endepunktet, jeg begynte med å opprette Stored Procedures for registrering og login, som legger til en bruker til databasen eller henter den inn ved login. Login endepunktet tar inn epost, authType (email, refreshToken), passord og refreshtoken. Her har brukeren mulighet til å velge å logge inn ved brukerinfo, eller refreshtoken. Dette gir brukeren mulighet til å lagre refreshtoken lokalt, og re-autentifisere med denne. Som jeg skal bruke for å slippe at brukeren blir logget ut av webappen hver gang den åpnes, eller etter auth token går ut. Refreshtoken varer i 7 dager, authtoken varer i 1 time. Prosessen ved innlogg er at brukeren hentes fra databasen via oppgitt info, en ny refreshtoken og authtoken generes, og data returneres. Data returnert er token, utløp for token og refreshtoken. Prosessen er identisk ved login av refreshtoken, men da blir refreshtokenen «fornyet» i stedet for opprettet.

Ved registrering så taas epost, fullt navn, og passord inn. Dette brukes da i en STored procedure for å opprette brukeren i databasen. Etterpå genereres en refreshtoken og authtoken likt som i login endepunktet, og returneres. Dette bidrar til at brukeren blir automatisk logget inn etter registrering.

Mye tid ble brukt på sikkerhet i koden, ved at jeg legger inn logging. Og god håndtering av exceptions, validaterings logikk på data osv.

Jeg la også inn en custom Authentication handler i APIet, ved bruk av ServiceCollections, som bruker mine egendefinerte tokens til authentication. Tokens blir kryptert med en privat RSA key, og blir validert med en offentlig RSA key.

Resten av dagen ble brukt på å optimalisere login og registrerings endepunktene, teste dem, rydde opp i koden og legge til swagger dokumentasjon på alle offentlige klasser.