1. **Iteration**

Client og Server
I vores projekt, bliver der på nuværende tidspunkt brugt nogle generelle netværksbegreber. Vores webserver har fx. en **IPv4 addresse** samt en **MAC addresse** på vores enhed, ligeledes har brugerne også en **IPv4 addresse** og **MAC addresse** på den enhed de benytter. Gennem disse to informationer, kan der kommunikeres mellem vores webserver og bruger ved hjælp af **TCP/IP (*OSI-modellen*)**, hvor data bliver genereret når det går gennem **OSI-modellens 7 lag (Encapsulation & De-encapsulation)** i takt med at det bliver leveret rundt ved hjælp af **Layer 2 (MAC-addressing scheme)** og **Layer 3 (IP-addressing scheme)** til forskellige enheder, og til sidst til enten vores webserver eller vores bruger.

2. **Iteration**

Direkte og indirekte protokoller der bliver brugt i vores projekt

Internet Protocol (IP)
er en logiske adresse for en bestemt enhed på et netværk, adresserne bliver uddelt af protokollen DHCP.

Dynamic Host Configuration Protocol (DHCP)
En DHCP server, står for at uddelegere IP-adresser til enheder på et netværk. Typisk så er disse IP-adresser kun til lån, dvs. at en enhed har kun en IP-adresse i en given længde, hvor den så skal anmode om at genlåne IP-adressen hvis den stadig er aktiv. IP-adresse kan dog også blive reserveret, dvs at DHCP sætter IP-adressen op mod enhedens MAC-adresse, så hver gang at enheden anmoder om at få en IP-adresse, så prøver den at give den samme IP-adresse igen. Dette er især vigtigt for fx en Web server som Kestrel og IIS, da IP adressen, specificeret på DNS serveren skal stemme overens, for at klienter kan sende requests til vores webserver.

Transmission Control Protocol (TCP)
er en protokol der skaber en sikker måde at transporterer data over et netværk, den sikrer at alt data bliver modtaget af den modtagende enhed, det gøres ved at den initiere et “Three Way Handshake” hvor der forhandles betingelser mellem de to enheder hvor der aftales f.eks størrelse på hver forsendelse af data i byte. håndtrykket bliver lavet ved at “host A” sender et SYN (synkroniserings sekvens nummer). til “Host B” som besvarer Den med et et SYN. samt et ACK (andkendese). for at angive hvilket SYn den forventer næste gang. til sidst sender Host A et ACK til Host B, derefter kan dataen blive sendt mellem de to hosts.

TCP er mest brugt til overførelse af data hvor det er vigtigt at alt data bliver overført uden tab f.eks Mail, Downloading/Overførsel af filer, og ved HTTP protokollen for at sikre at man får hele webstedets ressourcer.

men i tilfælde hvor det ikke er vigtigt at atl dataen bliver overført men hastigheden har større betydningen, f.eks ved videosamtaler og streaming services vil det være bedre at bruge UDP (User Datagram Protocol) som ikke laver et “Three ways Handshake”.

File Transfer Protocol (FTP) (måske er ikke sikker på om azura server bruger den eller om de bruger en anden protokol til overførsel af filer. Det er jeg 99% sikker på at de gør)
bruges til at dele overføre filer fra en host til en anden oftest i form af at overføre fra en computer til web server.

Hypertext Transfer Protocol (Secure) (HTTP(s))
er en protokol der kan hente ressourcer fra en kilde, den er hovedsageligt brugt på internettet til at kunne hente html dokumenter og css dokumenter og andre ressourcer fra et websted

fungerer også som mellemledet mellem .netværket og den server som webstedet er laget på.
Fordelen for http er at den er extensible, hvilket betyder at man kan nemt udvide på protokollen. Yderligere så bliver alt data transfer også krypteret, hvilket har gjort den til standarden for især forretnings hjemmesider.

3. **Iteration**

Overvejelser omkring implementering på flere servers

Man kan splitte sin hjemmeside op i dele, hvor hver del kører på sin egen server. Når en bruger så requester en specifik del fra index siden, så bliver requesten redirected (rerouted) til den server der håndtere den del. Man kan vælge en netværkstopologi som grundlag for den måde man sætter sin hjemmeside / servere op med. Her kunne man f.eks kigge på Mesh og Star pattern, til at sende requests og responses rundt.

Fordelene er at hvis én server går ned, så er de andre stadig oppe og køre.
Hvis en server går ned, så er den isoleret. Man kan derfor gå i gang med at fejlsøge. Alt efter størrelsen, kan det derfor være mere overskueligt at fejlsøge i sådanne tilfælde.

Ulemperne vil være at en request og response nok kan tage lidt mere tid for at komme frem og tilbage, hvis man har en “server-hub” der tager imod alle requests og redirecter til en anden server, for så at modtage og sende et response tilbage igen.
Der vil også være noget ekstra server tilsyn der skal foregå, da man nok kunne ende med at have mange servere kørende, alt efter hvor stor hjemmesiden er.

4. Iteration

SQL injections

Konsekvensen af SQL Injections er at ens database kan blive manipuleret således at sikkerheden brydes.
Vi bruger .Net Core som selv auto generer vores databases sql statements. .Net Core sikre mod sql injections ved at lave parmaticerede statements hvor det ikke er muligt at bruge SQL injection til at trække information ud fra databasen.

Beskyttelse af brugernes følsom data

problemet er hvis en person skulle få adgang til vores database og alt dens data, vil de havde fuld adgang til alt information omkring alle bruger der benytter vores service.

for at kunne formindske skaden der vil ske hvis nogen skulle få adgang til vores database. ville vi kunne hashe det følsomme data, for at kunne gøre dataen ulæselig , og eftersom hashing er envejs hvilket betyder at du kan ikke gå tilbage og se hvad det originale data var.dette går det optimalt til brug til kodeord eftersom at hashing er detamistisk hvilket garantier af resultat vil altid være det samme så når bruger logger ind vil der hashed kodeord passe med databasens værdi hver gang.
og få at gøre det mere besværlig kan vi tilføje en ekstra table til vores database som vil holde en tilfældig unik data kaldt salt, denne værdi tilføjes til kodeordet inden det bliver hashedet. dette gør det meget mere tidskrævende at finde frem til det originale kodeord.

Broken authentication & Broken Access control

Da vi kommer til at anvende et administrationsmodul, skal vi sikre at normale brugere ikke lige pludselig kan få administratoradgang. Vi skal ligeledes også sikre de funktionaliteter en normal bruger samt en administrator kan bruge. En bruger skal fx. ikke kunne se en oversigt over andre brugere og være i stand til at redigere i deres data. Så vi skal have noget regulering / kontrol over hvilke funktionaliteter de kan anvende.

Insufficient logging and monitoring

Logging af data, skal sikre at vi har noget at kigge og analysere på over længere tid, for at se om der har været nogle sikkerhedsbrud eller underlige forhold på vores hjemmeside / API. Vi skal sikre at vi har et middel hvorpå vi kan finde anomalier ifh til serverens tænkte funktionalitet, da sikkerhedsbrud ofte først kan ses efter flere måneder, ved netop at finde ting i ens logs, som ikke stemmer overens.