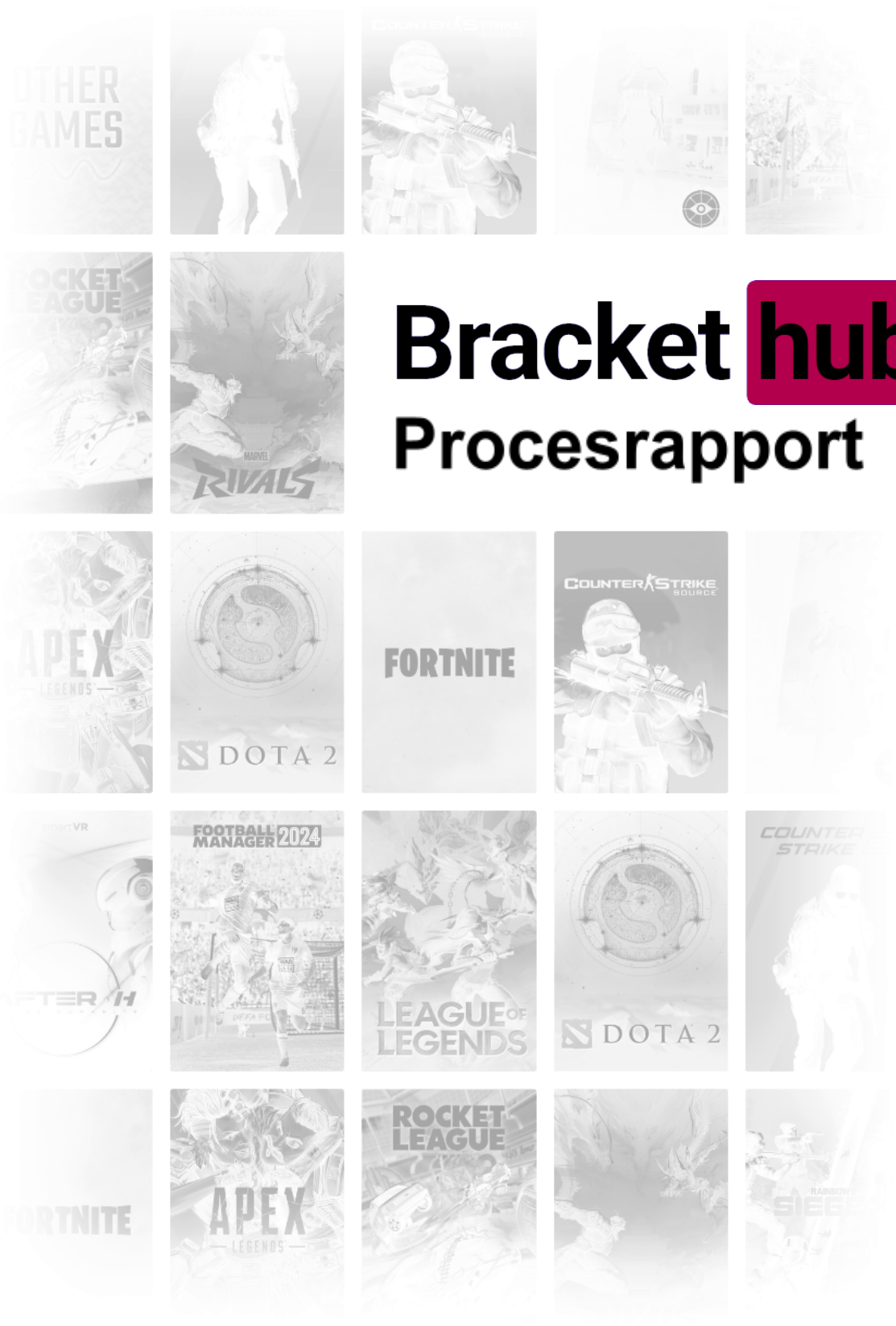

Bracket hub
Procesrapport

Titelblad

Information om processrapport

Deltager:	Julian Greger
Projektnavn:	Svendeprøve - BracketHub
Projektperiode:	15/09/2025 - 03/10/2025 (Aflevering)
Skole:	Aarhus Tech, Hasselager Allé 2.
Hold:	25hs37dat6pv1_1, - Data/It

Indholdsfortegnelse

Titelblad **1**
Indholdsfortegnelse **2**
1 Introduktion **3**
2 Læsevejledning **3**
3 Case **4**
4 Problemformulering **4**
5 Projektplanlægning **5**
5.1 Estimeret Tidsplan 5
Uge 1 5
Uge 2 5
Uge 3 6
5.2 Risikoanalyse 6
6 Metoder, Principper og Teknologier valgt **6**
6.1 Arbejdsmetode - Vandfaldsmetoden 6
6.2 Principper 7
6.2.1 SOLID 7
6.2.2 DRY 7
6.3 Teknologier 8
6.3.1 Blazor 8
6.3.2 HTML & CSS (SCSS) 9
6.3.3 C# - Frontend & Backend 9
6.3.4 SQL Server 10
6.4.5 Git (Github) 10
7 Udviklingsprocessen **10**
7.1 Datagrundlag 10
7.2 UI 11
7.2.1 Layout 11
7.2.2 Forside 11
7.2.3 Spil og Turneringer 11
7.2.4 Turnering 12
7.3 Backend 12
7.3.1 API 12
7.3.2 Database 12
8 Realiseret Tidsplan **13**
9 Perspektivering **13**
9.1 Videreudvikling (Mulig fremtid) 14
10 Konklusion **14**
11 Kilder **15**
12 Logbog **16**

1 Introduktion

Denne procesrapport skal være med til at forklare og fremhæve de tanker og udførelser, der har været i løbet af projektet. Rapporten gennemgår de metoder, principper og teknologier, der har været brugt til at udvikle projektet fra start til det endelige produkt.
Rapporten går igennem hvilke områder og elementer der er gået godt, samt det der er gået mindre godt. Den slutter så af med hvad der eventuelt mangler og hvad kunne have været lavet anderledes såvel som, en endelig konklusion på projektet.

2 Læsevejledning

Projektet består af 2 dele, en processrapport og en produktrapport. Denne sektion vil give et overblik over hvordan man skal læse de 2 rapporter samt hvilken rækkefølge der vil passe bedst til læserens preferencer.
Rapporterne er skrevet med udgangspunkt i at læseren minimum har lidt viden og forståelse inden for programmering, webudvikling og webdesign samt relevante termer.

Det anbefales at man læser processrapporten først hvis man interesserer sig for det tekniske, såvel som processen, da processrapporten er med til at forklare mere i dybden vedrørende tanker, valg of problemer.
Er man mere interesseret i designet af produktet så burde man læse produktrapporten først da den er med til at forklare hvad produktet går ud på såvel som hvordan forskellige elementer hører samen og ser ud.

Foklaring af jævnlige brugte termer:
Frontend og **UI**: Brugerflade (Det man ser når man bruger produktet)
Backend: Alt bag brugerfladen (Kode, services osv. som en bruger ikke vil se)

3 Case

Titel	BracketHub
Case Beskrivelse	Et community med fokus på gaming har udfordringer med organiseringen af turneringer på tværs af flere spil. De har siden starten altid brugt regneark og online chats til at holde styr på planlægning, hold og resultater. De ønsker sig et online system hvor folk kan tilmelde sig, holde styr på kampe og se hvordan det går for forskellige hold, både via web og mobil.
Teknologier	C# (Blazor)
	HTML
	CSS
	JavaScript
	Git
	SQL (SQL Server)
Fagområder	GUI-programmering
	Clientsideprogrammering
	Objektorienteret programmering
	Databaseprogrammering
	Versionering

4 Problemformulering

Hvordan kan communities sørge for at de undgår rod med turneringer og samtidig får et mere effektivt og overskueligt overblik. Hvordan kan man skabe et system, hvor medlemmer og intereserede nemt kan oprette, administrere og følge med i eksisterende turneringer, hold og resultater.
Løsningen skal være med til at give et visuelt klart overblik, og på samme tid gøre det nemmere for arrangørere og deltagere at gennemføre turneringer uden fejl og forvirring.

5 Projektplanlægning

5.1 Estimeret Tidsplan

	Uge 1.					Uge 2.					Uge 3.				
Emne	15	16	17	18	19	22	23	24	25	26	29	30	1	2	3
Projektplanlægning	■														
Procesrapport		■	■			■		■		■		■	■	■	■
Produktrapport				■	■		■		■		■	■	■	■	■
Opsætning - Git Repository	■														
Opsætning - Frontend	■														
Opsætning - Backend	■														
Package research	■	■													
GUI - Frontend	■	■	■	■	■				■		■				
API						■		■		■					
Database							■	■							
Test og forbedringer												■	■	■	

Projektet har strukket sig over 3 uger; d. 15. September til d. 3. Oktober.

Uge 1

Tidsrum: 15. September til 19. September.

Denne uge vil bestå af opsætningen af frontend og backend kode projekterne, samt opsætningen af et Git repository som kan bruges til at styre versioneringen af de 2 projekter. Udover den generelle opsætning så vil ugen bestå mest af frontend programmering i form af en hjemmeside for at få skabt noget visuelt hurtigt og få sat det op så at produktets backend kan testes og interageres med. Udover programmering så vil der være lidt rapportskrivning indimellem.

Uge 2

Tidsrum: 22. September til 26. September.

Uge 2 vil primært bestå af programmering af produktets backend, dette inkluderer en API og en database. API'et skal bruges af frontenden, så nu når der er lavet arbejde på frontend'en i uge 1, så er det oplagt at få sat backenden op. API'et vil ikke være så kompliceret så det der vil være mest arbejde i er at få dataen transporteret og vist ordenlig mellem frontenden og backenden, og så mellem backenden og databasen. Opsætningen og datastrukturen af databasen vil blive sat op i API projektet ved hjælp af C# koden. Der vil så udover koden også være lidt rapportskrivning.

Uge 3

Tidsrum: 29. September til 3. Oktober.

Uge 1 og 2 havde det største fokus på selve koden og produktet, så uge 3 vil primært være rapportskrivning såvel som tests og forbedringer af selve produktet. Produktet vil kunne tjekkes igennem for fejl og mangler, som så kan fikses og forbedres.

5.2 Risikoanalyse

Projektet vil bruge Blazor som er kendt for at være lidt speciel når angår opdatering af UI elementer med nyt eller frisk data. Hvis koden ikke bliver sat ordenlig op i forhold datahåndteringen mellem backend'en og frontend'en så kan det ske at UI elementer ikke bliver opdateret og kun viser ingen eller forældet data.

Projektet bliver bygget på at der skal være en forbindelse mellem data og UI, så i da projektet vil starte ud med at fokusere på UI delen, så kan det være svært at forholde sig til data og man bliver derfor nødt til at lave noget test-data lokalt som man så kan få lov til at teste UI'en men. Der vil være et simpelt diagram som kan følges vedr. udseendet og strukturen af den data der skal bruges, så man skal tage lidt højde for den aktuelle data struktur såvel som mulige rettelser der kan komme i forbindelsen med koden af UI'ens backend.

6 Metoder, Principper og Teknologier valgt

6.1 Arbejdsmetode - Vandfaldsmetoden

I projektet vil fremgangsmåden være vandfaldsmetoden. Vandfaldsmetoden er god i dette tilfælde, da den har en lineær struktur som passer godt til projektets fremgang.
Projektet har fået fastsat krav for hvad der skal laves, der er ingen "kunder" eller parter som kan eller har noget at sige vedr. produktet og produktet vil være udviklet af en enkelt person. Vandfaldsmetoden kan være med til at projektet får ramt alle faserne fra kravspecifikationen og frem til det endelige resultat af produktet.

I tilfælde hvor man har kunder eller andre parter som skal være med inde over processen fra start til slut, så kunne man have brugt en mere agil metode som f.eks. Scrum eller Kanban. Man kunne godt have brugt en af de mulige agile metoder i dette projekt, men det ville have været en selv som "kunde" som man så vil "spare" med i en cirkulær fremgang.

6.2 Principper

6.2.1 SOLID

Projektet kører på SOLID princippet if forhold til koden. Princippet består af 5 Principper:

1. ***Single Responsibility Principe***: En klasse skal kun have 1 formål.
2. ***Open-closed Principle***: Klasser skal være åbne for udvidelse uden modificering.
3. ***Liskov substitution principle***: Hver underklasse eller afledt klasse skal kunne substitueres af deres base eller overordnede klasse.
4. ***Interface Segregation Principle***: Klasser og interfaces burde ikke være tvunget til at implementere eller være afhængige af metoder som de ikke bruger
5. ***Dependency Inversion Principle***: Høj-niveau moduler bør ikke afhænge af lav-niveau moduler. Begge bør afhænge af abstraktioner.

SOLID princippet kan være med til at sørge for at turneringssystemet kan holdes ved lige, samt give muligheden for udvidelse. Ved at give hver klasse 1 ansvar, kan man sørge for at man kan tilføje nye funktioner uden at man skal modificere det eksisterende kode. Med små og fokuserede interfaces kan man gøre komponenter nemmere at bruge og teste. Ved at holde styr på modul niveauerne kan man få koden til at modtage nye formater og integrationer uden alt for store risikoer. Sammenlagt kan SOLID princippet give koden en mere flexibel struktur som kan sørge for at det understøtter udviklingen af turneringssystemet, dette gælder ikke kun for de nuværende krav men også for mulige fremtidige krav.

6.2.2 DRY

DRY står for “Don’t repeat yourself” og går ud på, at man prøver at holde kode så genenvendelig som mulig. Lidt ligesom KISS princippet for design, så kan DRY princippet sørge for at man kan holde ens kode enkel og vedligeholdbar.

Ved at genbruge så meget kode som muligt kan man spare tid og kræfter, imens man koder. Under udviklingen af forskellige systemer kan man ende ud i at skulle tilføje og rette i kode ud fra hvordan produktet går fremad. Sammen med SOLID princippet så kan man få lavet et robust system.

Eksempel af metode-genandvendelse i projektet:

```
public async Task<MemberModel?> MemberSignup(MemberCreateUpdateModel member, CancellationToken cancellationToken = default)
{
    string url = GetUrlWithQuery(nameof(MemberSignup));
    using HttpResponseMessage response = await httpClient.PutAsync(url, SerializeModel(member), cancellationToken);

    return await CheckAndConvertResult<MemberModel?>(response);
}
public async Task<MemberModel?> MemberSignin(MemberReadModel member, CancellationToken cancellationToken = default)
{
    string url = GetUrlWithQuery(nameof(MemberSignin));
    using HttpResponseMessage response = await httpClient.PostAsync(url, SerializeModel(member), cancellationToken);

    return await CheckAndConvertResult<MemberModel?>(response);
}
private async Task<T?> CheckAndConvertResult<T>(HttpResponseMessage response)
{
    if (CheckResult(response))
    {
        string responseString = await response.Content.ReadAsStringAsync();

        T? result = JsonConvert.DeserializeObject<T?>(responseString);

        return result;
    }

    return default!;
}
```

Koden viser 2 metoder som bliver brugt til henholdsvis “Signup” og “Signin”. De 2 metoder gør begge brug af en metode, som kan bruges til at omdanne et API svar om til en specifik model som så kan bruges i frontend delen.

6.3 Teknologier

6.3.1 Blazor

Blazor er et af Microsoft’s nyere kode frameworks der tillader full-stack webudvikling i kodesproget C#, HTML, CSS og JavaScript. Blazor udnytter komponent modeller sammen med et “cross-platform” udviklingsmiljø der bruger Microsoft’s .NET platform. Blazor gør det muligt at udvikle hjemmesider med logik bygget via C# i stedet for JavaScript. Javascript er standarden for den visuelle logik der kan bruges i hjemmesider (browsere er bygget til at håndtere kode som JavaScript sammen med HTML og CSS)
Blazor er bygget til at omskrive C# kode til f.eks. Javascript. Dette er med til at man kan bruge eksisterende C# færdigheder, biblioteker og logik.
Udover C# (og JavaScript) så udytter Blazor også HTML og CSS som er det som browsere bruger til at finde ud af hvordan elementer skal se ud.

Blazor kan bruges på 3 måder; Klient-side, Server-side og Hybrid. Klient-side også kendt som “Blazor WebAssembly” fokuserer på at alt kode bliver kørt i en bruger’s browser.
Så alt logik, API kald osv. vil køre hos brugeren, hvorimod server-side kan sættes op til at køre logikken, API kald osv. på en server hvorefter det så bliver sendt/streamet ned/hen til brugere’s browsere.

Projektet gør brug af Blazor WebAssembly da er nemmere at hoste og ikke vil bruge for mange ressourcer at holde kørende. Der er en lille forskel på hvordan man vil kode en klient-side hjemmeside frem for en server-side hjemmeside. Med klient-side skal man være mere opmærksom på, hvad man sender til klienten (API nøgler, “Secrets” samt andet hemmeligt data og information).

Man kunne sagtens have gjort brug af server-side, men da projektet ikke vil indeholde kompleks logik og opsætninger, så kunne der nøjes med WebAssembly.

6.3.2 HTML & CSS (SCSS)

HTML er et markupsprog der kan fortælle browsere hvilke elementer der er tale om. Nogle elementer har nogle specielle betydninger for hvordan en browser læser og viser dem. Ud fra hvilke HTML elementer der bliver brugt og hvordan de er sat op, så kan man få lov til at style elementerne med en hel række værdier og udseender.

HTML og CSS er en del af Blazor's udviklingsfremgang så det er mere eller mindre et krav at bruge på enten den ene eller anden måde. Det ville være meget svært at undgå brugen af HTML og CSS i et Blazor setup da de er en integreret del af web elementerne der bruges.

Et eksempel på HTML der møder C# (Muligt via Blazor's system)

```
@if (SortedGames != null)
{
    foreach (var item in SortedGames)
    {
        <a href="@TournamentList.NavigateMe(item.Type)" class="game-item">
            <img src="@ImageStatics.CoverUrl(item.Type)" />
            <span>@item.Name</span>
        </a>
    }
}
```

Blazor gør forbindelsen mellem C# og HTML mere eller mindre problemfri.
Eksemplet viser et stykke kode som bliver brugt til at vise et "link" element til hver spil i "SortedGames" listen.

6.3.3 C# - Frontend & Backend

C# er et moderne objektorienteret kodesprog (OOP - Object-Oriented Programming) som er kendetegnet ved at det er enkelt, fleksibelt og kan anvendes i mange områder. Kodesprogets princip ligger på, at alt koden er organiseret i objekter og klasser, hvilket gør at det er et godt kodesprog til moduler og skalering. C# er en af de kodesprog hvor der kan findes en masse frameworks og pakker, som nemt kan integreres uden alt for meget besvær, på samme tid med at der findes en masse information, guides osv. på nettet til situationer hvor man har brug for lidt hjælp.

C# er en af hovedepunkterne i Blazor da det er det kodesprog som Blazor bruger udover f.eks. HTML, så det var en selvfølge at C# ville være kodesproget til Frontend delen. Backend'en kunne laves på flere måder, da der er mange frameworks og kodesprog, som kan bruges til at lave f.eks. API'er og databaser. Men siden C# allerede er i frontenden og man også kan lave backends i samme sprog, så var det oplagt at bruge C# til API'et der skulle bruges til at snakke med både frontend og database.

6.3.4 SQL Server

I projektet er der gjort brug af SQL Server da SQL kan sikre datakonsistens, integritet og rationelle data forespørgsler, hvilket ville være essentielt for at kunne styre og vedligeholde spillere/teams, turneringer og kampe.
Med SQL kan man også sætte det op så man kan indexere data for maksimal præstation. SQL's struktureret skema er med til at gøre relationer mellem data simpler og nemmere at bruge.

6.4.5 Git (Github)

Valget af versioneringsværktøj har været Git sammen med Github. Git funger ekseptionelt godt med Visual Studio (IDE) via Github, da det er en af de versioneringsmetoder som Visual Studio understøtter ud af boksen.
Der findes andre værktøjer og hjemmesider der kan gøre det samme som Git og Github. Git er en af de nemmeste at bruge samt integrere med for at få et godt overblik over versioner af ens kode. Man kan både se hvordan kode har set ud, samt hurtigt gå tilbage til tidligere versioner hvis man formår at lave noget man fortryder.

7 Udviklingsprocessen

Projektet startede med at finde ud af, hvad der skulle bruges af pakker og frameworks til UI'en. Efter en omgang af research kunne der laves datamodeller som kunne bruges til som fundament over de UI elementer der skulle bruges i forhold til data.

7.1 Datagrundlag

Simpel oversigt over modellerne som er blevet udarbejdet til prototypen ud fra dens MVP (minimum viable product):

Game Model

Navn	Type
Name	string
Type	string
Description	string

Member Model

Navn	Type
Id	int (Index)
Name	string
Nickname	string

Match Model

Navn	Type
Id	int (Index)
Status	enum (int)
Round	int?
MatchNumber	int?
Winner	int?
Links	**Ref**
Members	Members - Id
ParentMatches	Matches - Id
ChildMatch	Match - Id

Tournament Model

Navn	Type
Id	int (index)
GameType	string
Status	enum (int)
Name	string
Description	string?
Date	DateTime
Links	**Ref**
Matches	Matches - Id
Members	Members - Id

Produktets MVP er relativt simpel i forhold til hvad der skulle være muligt i produktet. Hovedekravene for et funktionelt turneringssystem er som vist i oversigten; spil, medlemmer, kampe og turneringer. Modellerne har været bygget på, hvad de minimum skulle indeholde af information/data. Modellerne har haft holdt sig relativt ens iløbet af hele processen.

7.2 UI

Frontend delen var en af de første ting som der er blevet arbejdet på. Det startede ud med at der er blevet lavet mock-ups som så kunne bruges til at kode og designe de forskellige sider.

7.2.1 Layout

Layoutet på en hjemmeside kan have stor betydning for resten af hjemmesiden, så det var det første element der blev arbejdet på. Dette inkluderer top-baren som skulle indeholde hjemmesidens logo såvel som simple funktioner.
Topbaren er blevet designet til at tilpasse sig alle skærmstørrelser, så hvis man bruger hjemmesiden på en mobil eller anden mindre enhed hvor der ikke er så meget skærmplads, så sørger top-baren for at der kan bruges en burger menu til navigeringsknapper.
Selve hjemmesidens sektion med indhold for de forskellige sider er sat op til, at have en begrænsning for hvor stor den er i forhold til skærmstørrelsen der er i brug. Efter hjemmesidens layout var på plads kunne undersiderne sættes op.

7.2.2 Forside

Forsiden er som regel en af de vigtigste sider på en hjemmeside, da det er med til at skabe et førstehåndsindtryk hos brugere. Siden er blevet sat op med et søgefelt i toppen som vil kunne bruges til at søge efter spil og turneringer. Udover søgefeltet er der blevet sat et simpelt grid op med spil såvel som turneringer. Dette er med til at skabe lidt liv i siden for nye såvel som tilbagevendende brugere.
Forsiden er forblevet ens i dens layout efter første omgang, men der er i løbet af processen blevet tilrettet og tilføjet lidt visuel feedback af spil og turneringer, for at give dem lidt mere flair, end hvis det bare er flade billeder.

7.2.3 Spil og Turneringer

For at brugere kan få lov til at se alt hjemmesiden tilbyder, er der lavet 2 undersider til henholdsvis spil og turninger.
Både spil- og turneringssiderne henter data fra API'et for at vise aktuelle spil og turneringer.
Spil siden er designet til at være simpelt i designet i form af det samme grid som på forsiden, forskellen her er dog at forsidens spil grid er blevet ændret til at kun vise en håndfuld af spil, hvorimod spil siden viser alle spil som er tilgængelige i databasen.
Siden med turneringer er blevet sat op til at kun vise de spil som hører til den underside, så hvis man er gået ind for at se alle turneringer under spillet; "Counter-Strike 2", så er det kun turneringer fra det spil man kan se. Dette er med til at holde fokuset på de spil en bruger er interesseret i. Denne side har ikke haft nogen større ændringer i løbet af processen.

7.2.4 Turnering

For at give brugere information vedr. turnerninger, så er der lavet en underside ud fra spil og turnernings-id. Denne side er blevet sat op med information om den gældende turnering i toppen, så man hurtigt kan få et overblik over hvad den specifikke turnerning går ud på.

Turneringssiden henter data om turneringen via API'et som den så kan vise i et mosaik layout. Dataen der bliver hentet kan så bruges til at vise f.eks. medlemmer for turneringen, såvel som kampe der enten skal spilles eller har været spillet.

Til siden er der lavet et komponent som kan vise hvordan turneringens "bracket" ser ud. Dette komponent er et af de komponenter som har ændret sig mest i løbet af projektets process. Det har både været fordi at dataen ikke stemte overens med hvad der blev vist, men også fordi at daten der kom ind ikke var korrekt. Komponent blev i starten lavet ud fra statiske test-data som lå i koden, da API'et og databasen ikke var sat op på dette tidspunkt i processen. Efter API'et og databasen var oppe og køre kom der hurtigt bugs og problemmer frem i den måde som komponentet håndterede dataen.

7.3 Backend

Efter det meste af frontend'ens første omgang var færdig og sat op, er backenden blevet lavet. Backenden er i den mere simple ende af, hvad der skulle laves. Siden at der allerede var en forståelse samt modeller for hvad der skulle være af data, kunne processen med backend gå relativ hurtigt.

7.3.1 API

Det eneste API'et skal, er at fungere som mellemand mellem frontenden og databasen. Undersider der skal hentes data til er blevet lavet så der skulle laves endpoints som så ville kunne bruges til at oprette, hente og redigere data. Microsoft har gjort det nemt at få sat et API op, såvel som at få sat den op til at snakke med en database via deres .net framework. API'et er lavet ved hjælp af Microsoft's "AspNetCore builder" som ikke kræver meget opsætning for at virke. Ved at bruge "AspNetCore" kan API'et også laves, så det nærmest er plug and play via Microsoft's IIS Server som webapplikation.

7.3.2 Database

Databasen er blevet sat op via Microsoft's SQL Server hvor den bare er blevet sat op med de standarde indstillinger som kommer ud af boksen. I starten kunne API'et ikke få lov til at snakke med databasen via den "connectionstring" som databasen havde genereret, da den var sat op. Problemet var dog bare at der ikke var givet adgang til API'et. Efter der var givet adgang, så kunne databasen opdateres med en migrering fra API projektet for at opsætte de tilsvarende tabeller.

8 Realiseret Tidsplan

Type i forhold til estimeret tidsplan

- **Overlap**
- **Ny**
- **Manglende**

	Uge 1.					Uge 2.					Uge 3.				
Emne	15	16	17	18	19	22	23	24	25	26	29	30	1	2	3
Projektplanlægning	■														
Procesrapport		■	■			■		■		■	■	■	■	■	■
Produktrapport				■	■		■		■		■	■	■	■	■
Opsætning - Git Repository	■														
Opsætning - Frontend	■														
Opsætning - Backend	■														
Package research	■	■													
GUI - Frontend	■	■	■	■	■				■	■	■				
API						■	■	■		■					
Database						■	■	■							
Test og forbedringer									■			■	■	■	

Tidsplanen er blevet fulgt til i nogle områder mere end andre. Der har været nogle områder som har taget mere tid end forventet fordi at der har været lidt der drillede under udviklingen, f.eks. API og Databasen.
Rapportskrivning har ikke været fulgt så godt i starten af projektprocessen som forventet, så det har endt med at være noget der krævede mere tid over en kortere periode.

9 Perspektivering

I løbet af udviklingen af projektet har der været meget fokus på UI delene af hjemmesiden. En af elementerne som har haft taget mest tid i frontend-delen, var bracket komponentet. I stedet for at lave det fra bunden, så havde det nok været bedre at bruge et system der allerede var lavet, dette kunne være i form af en “package” eller “library” fra nettet.
Så det endelige produkt har haft mindre backend programmering hvilket også har været med til at forholdet fra funktionalitet til design ikke er så god. Selvom produktets MVP er blevet opfyldt, så kunne der sagtens have været mere fokus på backend-delen.

Selvom produktets hjemmeside har haft meget fokus på selve designet, så kunne det godt have haft mere personlighed. Ved hjælp af brugertests kunne man nok godt have fundet frem til et design og funktionalitet som vil passe bedre til målgruppen.

Under udviklingen af projektet har der været for meget brugt for meget tid på action og for lidt tid på reason. Udviklingsprocessen har ikke haft nok spørgsmål til hvorfor nogle elementer og funktioner er blevet lavet som de er. Havde man været mere kritisk overfor ens valg, havde det været muligt at få et mere sammenhængende resultat i forhold til frontend og backend.

9.1 Videreudvikling (Mulig fremtid)

Skulle man trække projektet videre, så ville f.eks. et reel login system være en af kravene for at få produktet ud i et produktionsmiljø. Man kunne f.eks. bruge OpenID til at tillade login for platforme som Steam (platform der indeholder spil). Man kunne også tilføje et selvstående login system men så ville man skulle sørge for at man overholder GDRP regler.

Et turneringssystem kan i mange tilfælde være et system som kan bruges til at holde styr på mange ting indenfor spil-verden. Så man skulle nok videreuvikle på bracket systemet for at give bedre information om nuværende, såvel som fremtidige kampe som brugere endten er tilmeldt eller er interesserede i.

Forbedring af API'et og databasen ville også være et godt område at få udviklet mere på hvis man skal overholde regler for ekstern adgang såvel som data integritet, da prototypen ikke har nok validering på hvad der bliver efterspurgt og hvad der kommer ind i af data i API'et.

10 Konklusion

Projektet er gået godt, hvis man tager udgangspunkt i at det har været et enmandsprojekt. Processen har haft nogle op- og nedture men er som udgangspunkt et vellykket projekt hvis man kigger på resultatet i forhold til opgaven.

Projektets mindste levedygtige produkt er er blevet opfyldt, men så heller ikke mere end det. Hjemmesiden har det som skal bruges for at man kan oprette og administrere turneringer såvel som nok funktionalitet for potentielle brugere til at opfylde kravene for produktet.

Produketet er med til at give et hurtigt og nemt overblik over spil og turneringer, så hvis man vil organisere eller bare være med i turneringer, så er det endelige resultat et godt valg. Hjemmesiden har et meget simpel visuel design og er med til at mindske rod med det rette antal funktioner. Produktets system er som udgangspunkt relativt simpelt, men tillader at man kan udvide det i fremtiden hvis man ville bringe produktet ud i den virkelige verden.

11 Kilder

Alle kilder har været tilgængelige og brugt mellem d. 15.September 2025 frem til d. 3. Oktober 2025 uden ændringer i indhold.

1. Inspiration til Tournerinssystem:
   https://www.hltv.org/

2. Inspiration til Events, Matches:
   https://www.hltv.org/events

3. Inspiration til Games
   https://steamcommunity.com/ (App på PC har “Library” tab med grid af Spil)

4. Reference til “Games”; Iconer, Bannere og Baggrunde:
   - 4.1. https://steamdb.info/app/730/info/ Counter-Strike 2
   - 4.2. https://steamdb.info/app/240/info/ Counter-Strike: Source
   - 4.3. https://steamdb.info/app/1422450/info/ Deadlock
   - 4.4. https://steamdb.info/app/2252570/info/ Footbal Manager 2024
   - 4.5. https://steamdb.info/app/252950/info/ Rocket League
   - 4.6. https://steamdb.info/app/2767030/info/ Marvel Rivals
   - 4.7. https://steamdb.info/app/359550/info/ Tom Clancy's Rainbow Six® Siege X
   - 4.8. https://steamdb.info/app/854110/info/ AFTER-H
   - 4.9. https://steamdb.info/app/1172470/info/ Apex Legends
   - 4.10. https://steamdb.info/app/570/info/ Dota 2
   - 4.11. https://steamdb.info/app/3632750/info/ Fortnite

5. FontAwesome (Free/Gratis ikoner)
   https://fontawesome.com/search?ic=free&o=r

6. Roboto font (Google font)
   https://fonts.google.com/specimen/Roboto

7. Microsoft SQL Server 2022
   https://www.microsoft.com/da-dk/sql-server/sql-server-downloads

8. Microsoft Cross-Origin Requests (CORS) documentation
   https://learn.microsoft.com/en-us/aspnet/core/security/cors?view=aspnetcore-9.0

9. Microsoft Entity Framework Core
   https://learn.microsoft.com/en-us/ef/core/

10. Microsoft DbContext/DbContextFacotry Lifetime, Configuration and Initialization
    https://learn.microsoft.com/en-us/ef/core/dbcontext-configuration/

12 Logbog

- 15/09/2025
  - Opsat tidsplan
  - Undersøgt Blazor templates
  - Opsat GitHub repository
  - Startet/Opsat Blazor og API projekter og lagt dem op med Git.

- 16/09/2025
  - Sat base “Page” op til navigation og indhold.
  - Fundet på et midlertidigt logo til projektet.

- 17/09/2025
  - Startet med at få lavet mockups (Wireframes) af de vigtigste sider.
  - Startet med forsiden; fået lavet søgefelt og “Games” grid.
  - Fundet forskellige billeder til en masse spil (Fundet via Steam som er offentligt tilgængelige. Tilladelse af ekstern brug er ikke bekræftet).

- 18/09/2025
  - Færdiggjort navigation til Mobil størrelser (Mindre header sammen med en udfoldelig menu).
  - Færdiggjort navigation PC størrelse (Større header med mere indhold da der er mere plads).

- 19/09/2025
  - Fået lavet et “grid” til tornaments.
  - Startet med Tournament side som kan vise tournament indhold.
  - Fået lavet et bracket system til at vise kampe i en turnering.

- 22/09/2025
  - Rafineret “Games” og "Tournament"-grids.
  - Opdateret CSS Stylesheet.

- 23/09/2025
  - Ryddet op i filstrukturen.
  - SQL forbindelse og “Entities” oprettet og sat op i API projektet.
  - Tilrettet UI håndtering af data.

- 24/09/2025
  - Tilrettet “Entities” og oprettet Migrations.
  - Tilrettet Models for at passe bedre til Entities.
  - Oprettet Shared projekt for så modeller kan bruges af både frontend’en og backen’en.
  - Opdateret CSS Stylesheet

- 25/09/2025
  - Opdateret EFCore version for at undgå fejl med Migration.

    - Opdateret Connectionstring til SQL.
    - Fixet DB Entities for at minimere datastørrelse og forbedre queries.
    - Lavet ny migration til databasen.

- 26/09/2025
    - Opdateret til DbContextFactory i stedet for normal DbContext
    - Opdateret forbindelsen til API'et fra UI'en.
    - Opdateret API Klientet for UI'en

- 27/09/2025
    - Tilføjet et brugerkomponent til at vise bruger info.
    - Tilføjet vindue til "Signup" og "Signin".
    - Tilføjet Modeller til "Signup" og "Signin".
    - Opdateret bruger knappen i top-baren.
    - Tilføjet separat side til redigering af turneringer.
- 29/09/2025
    - Tilføjet "favicon" ikon til hjemmesiden.
    - Tilrettet undersider med titel som bliver vist i browser tabs.