**Forside (se afsnit 2.1)**

Forside med projektets titel, forfatter(e), institution og evt. andre relevante oplysninger.

League of Legends was officially released on October 27, 2009. It was developed and published by Riot Games and has since become one of the most popular and enduring MOBA video games in the world. Following this growth

In 2020, University College Lillebælt (UCL) implemented a new system for recording student well-being and the institution’s possible deficiencies. This system was set in place so that management could work more analytically and thereby attack UCL’s shortcomings more methodically. As the system grew, more values were added, and the registrants of the system began to grow annoyed with the many clicks and pages they had to go through. This report has therefore been developed in conjunction with a project that seeks to simplify and optimize the supervisor’s use of the registration system so that they can get through it faster and easier. After results gathered from several user interviews and a thorough analysis of the new system, the course for the project was set. The project would specifically look at how the system could be optimized without reducing the data sent and an attempt would be made to implement an internal CRUD-system to handle the data easily and with more visual clarity. To bring the idea of the new solution to life, several prototypes were developed which were tested and revised before a final design was determined. The web project has made use of PHP and SQL for the handling of data, and has, in hopes of speeding up the use of the system, made buttons and input fields easier to access by using design principles. To further speed up the use, the solution makes use of preselected values to reduce the demands of actions on the users part. This solution has since been proven through testing to be user-friendly and promote speed in registration as it encapsulates and solves the pain points that they expressed in the earlier test phases.

**Indholdsfortegnelse (se afsnit 2.4)**

Indholdsfortegnelse

[3.1 Indledning 6](#_Toc157165656)

[3.1.2 Problemdefinition eller problemformulering 7](#_Toc157165657)

[3.1.3 Afgrænsning 8](#_Toc157165658)

[3.1.5 Metodeliste 9](#_Toc157165659)

[Begrebsliste: 10](#_Toc157165660)

[Discovery: 11](#_Toc157165661)

[Dataindsamling: 11](#_Toc157165662)

[RIOTGAMES API: 11](#_Toc157165663)

[Datadragon: 11](#_Toc157165664)

[Konkurrent research: 11](#_Toc157165665)

[League Of Graphs: 12](#_Toc157165666)

[Lol Esports: 12](#_Toc157165667)

[Interpretation: 13](#_Toc157165668)

[Målgruppe: 13](#_Toc157165669)

[Persona: 14](#_Toc157165670)

[UI inventory (User patterns): 14](#_Toc157165671)

[Ideation: 14](#_Toc157165672)

[Prototype: 14](#_Toc157165673)

[Informationsarkitektur: 14](#_Toc157165674)

[Experimentation: 14](#_Toc157165675)

[VueJS (Framework): 14](#_Toc157165676)

[Udviklingen af platformen: 14](#_Toc157165677)

[”Spil-bibliotek” (Database): 14](#_Toc157165678)

[Nunubot: 14](#_Toc157165679)

[Key NodeJS Pakker: 18](#_Toc157165680)

[Axios 18](#_Toc157165681)

[Evolution: 18](#_Toc157165682)

[Statistik side 18](#_Toc157165683)

[Forside 18](#_Toc157165684)

[Helhedsoplevelse 18](#_Toc157165685)

[Rengøring af kode 18](#_Toc157165686)

[Konklusion: 18](#_Toc157165687)

[Perspektivering: 18](#_Toc157165688)

**Evt. læsevejledning**

En kort vejledning til læseren om, hvordan dokumentet er struktureret og hvordan det bør læses.

3.1 Introduktion

# 3.1 Indledning

League of Legends' eksplosive vækst har ført til en betydelig udspredelse af information og data inden for RIOTGAMES’ spilunivers og internettet. For League of Legends-entusiasten, er konstant vedligeholdelse og forbedring af deres spilfærdigheder alt afgørende for at kunne klatre ranglisten og derved opnå en berigende spiloplevelse. Desværre er mange af informationerne, der kan føre til forbedring af færdighederne hos den gældende spiller, ofte opdelt og spredt over flere forskellige hjemmesider. ”Organiseringen” af denne information, der omfatter statistik, nyheder, ”build-guides” og professionelle opdateringer, udgør en betydelig udfordring for spillere, der stræber efter være opdaterede og forbedre deres færdigheder.

Bachelorprojektet, som denne opgave præsenterer, under navnet ”LOLSTATS”, udspringer af behovet for en centraliserende platform. En platform, der ikke blot aggregerer og samler den abnorme mængde data der findes omkring spillet, men også præsenterer den gennem en struktureret og brugervenlig grænseflade. Målet for øje vil være at levere en helhedsorienteret oplevelse for League of Legends-spilleren. En platform, der kombinerer statistik, ”build-guides” og levere professionelle nyheder fra e-Sportsscenen. Ved at integrere nyheder og videostreams af professionelle spil på platformen skabes der ikke kun en omfattende informationskilde, men også en levende og dynamisk spil-fællesskab. Tanken er at spillere vil være i stand til at holde sig opdateret med de seneste begivenheder og lære af professionelle spilleres strategier via disse livestreams og dermed opnå en dybere forståelse af spillet og dets konstante udvikling. Du vil igennem denne rapport kunne læse hvordan denne platform er blevet udviklet og hvilke metoder der har været taget i brug og herved hvilke begrundelser der ligger bag specifikke tiltag taget.

# 3.1.2 Problemdefinition eller problemformulering

Efter du har skrevet en god indledning/problembaggrund, skitserer du den/de problemstilling(er), som opgaven omhandler. Med udgangspunkt i den valgte problemstilling skal du efterfølgende formulere din problemformulering/problemdefinition (brug kun et af disse ord – i opgaveoplægget vil det fremgå hvilket). Problemformuleringen/problemdefinitionen skal være kort, entydig og præcist beskrive det, du vil undersøge i din opgave. En problemformulering/problemdefinition består af et overordnet undersøgelsesspørgsmål og kan suppleres med underspørgsmål. I nogle skriftlige opgaver får du opgavebeskrivelsen direkte fra din underviser, og du skal derfor ikke skrive en problemformulering/problemdefinition selv. I disse tilfælde indsætter du opgavebeskrivelsen i stedet for en problemformulering/problemdefinition, så du stadig gør det klart for læseren, hvad du vil beskæftige dig med. HUSK: Du skal altid besvare hele din problemformulering/problemdefinition i opgavens konklusion.

Hvordan kan LoL-entusiastens udviklingsmuligheder bedst præsenteres på brugergrænsefladen og hvilke teknologier kan assistere til at optimere bearbejdelsen af den abnorme mængde data?

Underspørgsmål:  
Hvordan kan brugergrænsefladen skabe øget engagement og derved vedligeholde brugere?

På hvilken måde kan dataene fra brugergrænsefladen fremvises overskueligt?

Hvordan kan LOLSTATS skabe nok værdi til at bruges over konkurrenterne?

# 3.1.3 Afgrænsning

Efter du har skrevet din problemformulering/problemdefinition eller indsat opgavebeskrivelsen, er det vigtigt, at der kommer et afgrænsningsafsnit. I afgrænsningsafsnittet skal du således beskrive de emner som kunne have været inddraget i opgaven, men som du har valgt at se bort fra. Husk at begrunde dine fravalg. F.eks.: Grundet virksomhedens størrelse, afgrænses det til at tage udgangspunkt i den afdeling, som problemet vedrører.

Afspejlet i den følgende rapport vil den opmærksomme læser bemærke at projektet har lagt særligt fokus på spilleroversigten da det er her nøgleinformationer findes for entusiasterne der søger efter udvikling indenfor deres færdigheder. Valget af dette fokus er blevet truffet i overensstemmelse med min vejleder for projektet. Da LOLSTATS som en fuldendt platform er et overvældende stykke arbejde for en enkelt webudvikler at opbygge over projektets periode. Herfor er formålet med denne afgrænsning at målrette er at skabe klarhed i projektet ved at koncentrere sig om et specifikt område, som vejlederen har identificeret som væsentligt og relevant for projektets formal, samt muligheden for at kunne fremvise projektdeltagernes tekniske kunnen indenfor faget. Denne afgrænsning indebærer derfor, at andre potentielle områder eller funktioner bevidst er blevet udeladt eller udsat til senere overvejelse/udvikling for at sikre en repræsentativ og målrettet tilgang til udviklingen af platformen.

# 3.1.5 Metodeliste

Design Thinking (IDEO, 2012, s. 11):

Formålet med Design Thinking er, at omskabe besværlige udfordringer til

konstruktive designmuligheder samt skabe bedst muligt flow i projektarbejdet.

Denne proces er ikke kun brugt I udviklelsen men også som skabelon til rapporten for at fremme oplevelsen hos læseren.

Persona (Interaction Design Foundation, 2020):

Udformningen af en persona skal bidrage til at tydeliggøre, hvilken

målgruppe ens klient, og dermed projekt, sigter efter at ramme.

Brugertest (Saffer, 2010, s. 181-85):

Brugertest er en række metoder til at måle, om man med sit produkt/koncept

er på vej i den rigtige retning. Ved at lytte til brugeres feedback, kan man

korrigere sit design, således at det rammer målgruppen.

Prototyping (IDEO, 2012, s. 58):

Prototyper er en funktionel, men ikke komplet udgave af ens

produkt/koncept. Disse bruges for at teste, om funktionerne virker, som de

skal. Denne metode anvendes til fremvisning samt til bruger testning af

produktet.

Primær- og Sekundær research, Data indsamling (Svendsen, 2015).

Med primær og sekundær research, er der blevet indsamlet data til projektet. Som primær research er der både udarbejdet kvalitativt og kvantitativt data. Primær research er indhentet ved interview mens sekundær research er indhentet ved undersøgelse af konkurrenter.

Interviews (Higgs, 2020):

Ved at udføre interviews kan man få adgang til mere dybdegående

informationer omkring ens emnefelt. Interviews kan bruges som det kvalitative

modstykke til spørgeskemaer og andre kvantitative researchmetoder.

BMC (Businessbuddy.dk, 2021)

Business model Canvas er en strategisk model som kan bruges til at

dokumentere et firmas nuværende forretningsmodel. I dette projekt er modellen

brugt til at skabe en større virksomhedsforståelse samt hvilken værdi der kan

tilføjes til deres eksisterende forretningsmodel.

Database Normalisering – Normal former (studytonight.com, 2021)  
Databasenormalisering er en teknik til at organisere data i databasen. Normalisering er en systematisk tilgang til nedbrydning af tabeller for at eliminere redundans og er derfor blevet brugt til at opsætte en optimal database til projektet.

NoSQL Database model Diagramming (techighness.com)  
NoSQL data model diagramming er blevet brugt til at udvikle og planlægge en database ved at visualisere strukturen og forholdene mellem datainput i en NoSQL-database. Diagramet har giver et overblik over, hvordan dataen organiseres og gemmes i databasen uden at følge det traditionelle relationelle databaseforhold.

# Begrebsliste:

For at opnå den bedste forståelse af den følgende rapport er der udviklet en begrebsliste der kort forklare de forskellige termer der gøres brug af i online fælleskaber omhandlende League of Legends som platformen appellere til:

1. **LoL**: Det primære akronym brugt til at referere til League of legends
2. **Champions:** Spilbare figurer i spillet, hver med sine egne unikke evner og rolle.
3. **SOLO/DUO:** Den primære spilmetode i League of Legends, hvor holdene kæmper om at ødelægge modstandernes Nexus.
4. **Lane:** De tre hovedstier (Top, Mid, Bot) på Summoner's Rift, hvor kampe finder sted.
5. **Jungle:** Det område mellem lanerne fyldt med monstre; junglen er ansvaret for at nedbryde monstre og hjælpe lanerne.
6. **CS (Creature Score):** Antallet af dræbte fjendtlige minions (creeps); det bruges til at måle en spillers evne til at samle guld.
7. **Smurfing:** Adjektiv der refererer til en spiller som spiller på en alternativ ”account” der er rangeret lavere end hvad spilleren normalt er.
8. **Main:** Spillerens normale bruger, kan alternativt referere til brugerens højst rangerede ”account” hvis de har flere
9. **Buff/Nerf:** Positive eller negative effekter, der påvirker en champions evner og-/eller statistikker.
10. **Objective:** Et kraftfuldt monster, hvis drab giver holdet en permanent fordel.
11. **Ultimate:** En champions mest kraftfulde evne, ofte låst op på niveau 6.
12. **Cooldown:** Den tid, det tager for en evne at blive brugbar igen efter anvendelse.
13. **Meta:** Den aktuelle metode eller strategi, der anses for effektiv i spillet på et givent tidspunkt.
14. **Vision/Warding:** En synlige enhed, der giver synlighed og fjerner fjendens synsfelt.
15. **RIOTGAMES:** Det firma der står udvidelsen og vedligeholdelsen af League of Legends.

Disse begreber er grundlæggende for forståelsen af League of Legends, og er for spillere samt læsere af denne rapport vigtige at være fortrolige med for at øge forståelsen af rapportens indhold.

# Discovery:

## Dataindsamling:

Da dette projekt tager udgangspunkt i at være en aggregations platform er det første trin i designprocessen at tilegne sig viden om hvilken data der er tilstedeværende på diverse medier.  
Efter at analysere de fremtrædende datakilder hvorfra det ville være muligt at hive data fra blev der fremfundet 3 hovedkilder hvorfra langt det fleste LoL-entusiaster drager deres viden fra. Den første blandt disse er League of Legends hjemmeside hvorfra man finder generelle data omkring spillet og mere specifikke nyheder samt historier vedr. de forskellige champions der findes i spillet. Det næste sted hvorfra der kan findes en masse data omkring den professionelle scene er Leaguepedia som er en community drevet side hvor alle kan søge om rettighed til at redigere sider og derved er denne side ofte den hurtigste primære datakilde entusiaster søger til når de vil vide noget specifikt inden for scenen. Ved undersøgelsen af denne informations kilde blev der desvære fundet frem til at LOLSTATS ikke ville kunne gøre brug af denne data da deres data er brugerdreven og ikke gør brug af APIer.

## RIOTGAMES API:

Til sidst er der den primære datakilde som alle platformens konkurrenter gør brug af. Langt størstedelen af den relevante data de konkurrerende statistiksider bruger til udviklingen af deres platforme kommer fra RIOTGAMES’ API, som herfra vil omtales RGAPI. RGAPI refererer til en kollektion af API’er udgivet af RIOTGAMES, og giver brugeren adgang til en abnorm mængde data der indsamles omkring alle spil udspillet på deres officielle servere. Ved at ansøge om en APIKey til denne kollektion opnås muligheden for at kunne indsamle dataene om alle spillede spil. Desværre giver denne api ikke adgang til alle spil udspillet gennem sæsonen, eller rettere at sige, der gives ikke overskuelig adgang til denne data. RGAPI giver adgang til alle spillede spil men for at kunne hente den givne data ned skal brugeren angive et account id og gives derefter en historik på alle spil den givne spiller har spillet. Dette gør at LOLSTATS må oprette en sekundær database til indsamling og behandling af data hvis platformen ønsker at lave nogen form for behændig statistik.

Datadragon:  
Sammen med RGAPI udlover riot også en companion ressource navngivet: ”Riot Data Dragon” som er en samling af opretholdte ressourcer der anvendes i League of Legends. Data Dragon fungerer som en omfattende database og arkiv, der indeholder information om tekstdata, champions, runer, ikoner, billeder og andre relevante informationer i spillet. Ved at gøre brug af denne ressource for lolstats adgang til de interne referencer der findes i RGAPI til at lokalisere diverse assets der kan refereres til.

## Konkurrent research:

I takt med dataindsamlingen er der blevet gjort en grundig research i hvordan konkurrenterne præsentere deres data og deres platform/brand. I denne fremfunden blev der gjort opmærksom på de stærkeste konkurrenter. Disse konkurrenters stærke og svage sider vil ses her:

U.gg’s brugergrænseflade er meget brugervenlig og intuitiv, specielt for spilleren velbekendt med de forskellige termer og begreber der gøres brug af inden for scenen. I modsætning til nogle af konkurrenterne er u.gg’s indhold dog meget mere generaliseret og der fokuseres oftest på statistikken og ikke så meget på forklaringen af alle de underliggende billeder/termer/strategier. Herudover udlover u.gg, mod betaling, I samarbejde med højt rangerende spillere, supplerende video materiale til at illustrere strategier og kombinationer af angreb som den erfarne spiller kan lære af.

U.gg er en meget velkendt statistik side på markedet og for at kunne konkurrere med denne komplette platform vil dette projektet drage imod at fange de brugere der frafalder u.gg på deres svage sider, som oftest tager form af ”community engagement”. Der er simpelthen en mangel på fremhævede data der lader spilleren fremvise deres præstationer. Spillerens tidligere rangering, same overordnede statistikker menes ikke at være nok I fokus under interview med erfarende spillere (quote).

### League Of Graphs:

Grundet i at alle konkurrenterne får deres data fra den samme centraliserede kilde, RGAPI’en, ses der en trend blandt den fremviste data og måden den bliver præsenteret på mange af de konkurrerende platforme. Dette er også gældende på denne udvalgte konkurrent, League of Graphs. League of Graphs deler mange ligheder med u.gg men udskiller sig ved gå udover data generaliseret data fra spillet og har opbygget et kæmpe bibliotek af assets brugt gennem tiden af riot i spillet. Hvilket har resulteret i den omvendte effekt af u.gg. Hvor u.gg har lige den rette mængde data til at tilfredsstille den almene spiller, har League of Graphs så dybdegående statistikker at det kun er de mest teori interesserede brugere der gør brug af platformens mange resurser.

Håbet er at dette projekt kan fange de brugere der føler sig intimideret af den store mængde data League of Graph fremstiller, uden at opgive datafremstillingens integritet, ved at supplere en platform der kan finde den rette mængde data at fremstille på et givent tidspunkt. Herved vil brugeren have mulighed for at kunne opsøge data omkring builds, items og champions på deres hånd uden at ”føle sig angrebet af dataen” (quote).

### Lol Esports:

Lolesports kan være farlig at se som en konkurrent fremfor samarbejdspartner da dette er RIOTGAMES egen platform til fremvisning af live events, og vil til hver en tid have muligheden for at nedlukke platformens konkurrencemæssige evner ved at udelukke brugen af deres API. Dog bør siden alligevel ses som en konkurrent i dette projekts søgen på at supplere højere brugerengagement, da dette er siden spil-fællesskabet flokkes til når stemningen er højest. Under research perioden er det blevet klargjort engagement metrikkerne er højest når der fremvises professionelle spil og spillerne for mulighed for at deltage i de mange forskellige former for gamification herom som RIOTGAMES tilbyder.

Videostreams fører ofte en stigning I opholdstid (dwell time) på platformen som kan lede til forøget indtjening via annoncer og at brugeren vil komme tilbage til hjemmesiden. Herudover vil platformen også kunne tage brug af forskellige former for gamification som, lolesports gør, for at øge bruger registrering samt sidens delingsfrekvens.

# Interpretation:

## Målgruppe:

Efter at have analyseret konkurrenterne på markedet samt den kundegruppe platformen skal henvende sig til, er der blevet udført en målgruppe analyse for at kunne udvikle den gennemsnitlige bruger profil i en persona.

1. **Første Prioritet: Aktive League of Legends-entusiaster**
   * **Behov**: Disse brugere søger detaljerede statistikker, strategier, og guideoplysninger for at forbedre deres spil. De er interesserede i at forstå spillets meta, de bedste champions og item builds.
   * **Sprogbrug**: De bruger ofte spillets jargon og forstår komplekse spilbegreber. Kommunikationen er ofte direkte og fokuseret på spillets aspekter.
   * **Værdiskabelse**: Værdien for disse brugere ligger i at give præcise, opdaterede og dybdegående analyser og råd, som direkte kan anvendes til at forbedre deres spiloplevelse og præstationer.
   * **Aldersgruppe**: Typisk i alderen 16-30 år. Denne gruppe har ofte mere tid og engagement til at dykke ned i spillets detaljer og forbedre deres færdigheder.
2. **Anden Prioritet: Casual League of Legends Spillere**
   * **Behov**: Casual spillere er interesserede i generel vejledning og tips til at nyde spillet mere. De søger måske også at forstå spillets grundlæggende uden at dykke for dybt ned i komplekse strategier.
   * **Sprogbrug**: Mere afslappet og mindre teknisk end hardcore spillere. De foretrækker simpel og letforståelig information.
   * **Værdiskabelse**: For denne gruppe er det vigtigt at tilbyde letfordøjelige guides, sjove spilfakta, og generelle tips, der gør deres spiloplevelse mere behagelig.
   * **Aldersgruppe**: Aldersmæssigt mere varieret, ofte i alderen 15-35 år. Casual spillere omfatter yngre spillere, der lige er begyndt at spille, samt ældre spillere, der måske har mindre tid eller interesse i at fordybe sig i spillet på et avanceret niveau.
3. **Tredje Prioritet: E-sports Entusiaster og Fans**
   * **Behov**: Denne gruppe er interesseret i e-sportsscenen omkring League of Legends. De søger information om professionelle spillere, hold, turneringer og analyser af professionelle kampe.
   * **Sprogbrug**: En blanding af spiljargon og almindeligt sportsjargon. De er interesserede i både spillets tekniske aspekter og de bredere narrativer i e-sport.
   * **Værdiskabelse**: At tilbyde opdaterede nyheder, dybdegående analyser af professionelle kampe, og information om e-sportsbegivenheder kan tilfredsstille denne gruppes behov.
   * **Aldersgruppe**: Typisk 18-35 år. Denne gruppe omfatter unge voksne, der følger e-sport som en hobby eller passion, og de, der er vokset op med e-sport som en del af gaming kulturen.

Som ses i analysen overlapper en stor gruppe af målgruppe under den samme alder, køn og sprogbrug. Dette resulterer i at det bliver nemmere at nå ud til brugerne. Det overlappende sprogbrug gør det nemt at kommunikere til alle 3 målgrupper uden at noget går tabt blandt de forskellige spil termer, begreber og ikoner. Herudover fører en overlappende aldersgruppe ofte til øget engagement blandt grupperne når de mødes online. Dette kan være som følge af den mulige gamification der blev diskuteret i forrige segment. Herudover resulterer det også i en meget nemmere markedsføring da målgrupperne alle har spillet som interesse og ligger i ca. samme aldersgruppe.

## Persona:

For mere målrettet at kunne identificere målgruppen er der blevet dannet to personaer, Christian Pedersen og Alex Jensen (Bilag 1, 2). De to personaer er skabt til at give en dybere forståelse af de før opstillede målgrupper platformen henvender sig mod. Ved at udvikle disse personaer bliver det muligt at nemmere og mere effektivt tilpasse indhold og funktionaliteter mod de specifikke behov interesser og præferencer som karakteriseres i disse målgrupper. Dette er et vigtigt skridt mod at sikre at platformen resonerer med de tilvalgte målgruppers forventninger, hvilket ofte leder til øget engagement, tilfredshed og loyalitet.

Alex Jensen repræsenterer de dedikerede og engagerede spillere, som virkelig stræber efter at forbedre sig og opnå den højst mulige rangering for deres evner. Denne gruppe er interesseret i dybdegående analyser og detaljerede data der kan hjælpe dem med at mestre spillet på et dybere niveau. Disse spillere værdsætter nøjagtighed og relevansen af dataene når det gælder de evigt forandrende balancerings opdateringer vedrørende league of legends.

Christian Pedersen repræsenterer derimod de casual spillere, som værdsætter en mere afslappet og underholdende spiller oplevelse uden nødvendigvis at dykke ned i de dybere lag af spilstrategi og optimering. Denne målgruppe søger indhold, der er lettilgængeligt og forståeligt, hvilket hjælper dem med at nyde spillet mere uden at kræve en betydelig investering i tid og ressourcer.

Ved at skabe og anvende personaer som Christian og Alex, kan udviklere og marketingfolk bag projektet målrette deres tilbud og kommunikation mere præcist. Det gør det muligt at differentiere indholdet, så det appellerer til både casual og dedikerede spillere, sikre relevante brugeroplevelser og øge chancerne for succes på markedet

# Ideation:

## Prototypen:

Hjemmesidens prototype er udviklet i Adobe XD. XD er et populært værktøj blandt UI designere der gør det muligt at skabe high-fidelity designprototyper. Skabelsen af disse prototyper gør det muligt at simulere og test bruger interaktioner og udforme navigationsflows på hjemmesider eller apps før den egentlige udvikling begynder. Prototypen udviklet specifikt til denne platform har fungeret som en visuel guide og visualisering af platformens interaktions skelet for den endelige kodeudvikling.

Da platformens prototype blev skabt, forhenværende projektets startdato, under et andet forløb, har den tjent som grundlagsramme for projektets webudvikling, og mens den i dens tid undergik detaljerede udforskning og uddybende designbeslutninger, vil denne proces ekskluderes denne rapport da den vægtes til at have mindre relevans for udviklingsprocessen i det givne udviklingsforløb.

Konsekvensen af at have udviklet hjemmesidens prototype på forhånd betyder, at rapporten i stedet vil fokusere på andre aspekter af projektet, såsom implementeringsteknikker, udviklingsudfordringer, og evaluering af brugerfeedback. Dette tillader at dykke dybere ind i de tekniske og analytiske aspekter af projektet, frem for at opholde sig ved de indledende designovervejelser, som prototypen blev udviklet til at adressere i et tidligere forløb.

# Experimentation:

## VueJS (Framework):

Til udviklingen af platformen er der blevet brugt VueJS. Vue.js er ofte rost for sin enkle og lette læringskurve samt dets let vægtige framework der gør det nemt at vedligeholde og øger effektiviteten af koden da der ikke inkluderes en masse indbyggede funktioner der ikke er brug for. Det er et komponentbaseret framework gør brug af template-baseret struktur. Denne tilgang tilbyder fantastisk fleksibilitet som gør vue godt velegnet til udvikling af små til mellemstore web-projekter.

Som beskrevet er vue et komponent baseret framework og dette bringer mange fordele, såsom genanvendelighed, skalerbarhed, vedligeholdelse og læselighed. Genanvendeligheden ses i at komponenterne kan kaldes flere forskellige steder på applikationen på en gang. Skalerbarheden gir sig udtryk i nemt at kunne fjerne og /eller tilføje komponenter til andre filer. Dette betyder at applikationen kan udvikles og vokse uden at man mister overskueligheden, men også at komponent udvikling kan uddelegeres uden at forulempe andre ved at overskrive data, variabler, styling når koden samles da det alt vil være isoleret til den ene komponent. Hertil ses også vedligeholdelses fordelene da problemer nemt kan identificeres og isoleres. Til sidst nævnes læseligheden, der ses ved at koden bliver opdelt i mindre håndterbare dele, som gør det mere overskueligt.

Projektet har selvfølgelig gjort brug af disse komponenter i udviklingen af platformen og der vil senere beskrives hvordan der gøres brug af forskellige metoder og operationer vueJS stiller til rådighed som framework.

Endnu en grund til der er blevet valgt at arbejde med vue er pga. det indbyggede state-management system kaldet ”pinia”. Pinia biblioteket gør det muligt at dele tilstande, funktioner og variabler, på tværs af alle webplatformens sider og komponenter. Dette simplificere en masse udviklings problemer der ellers kan forekomme med funktionaliteter der behøver

I projektet er der anvendt Vue.js komponenter. Komponenterne er opdelt i to hovedkategorier kaldet cards og UI.

Eksempelvis er der lavet en template for et category card, som er cards på forsiden, og herefter var det muligt med Vue.js at lave en v-for, som genererer et card for hvert data objekt fra Firebase. På den måde vil der nemt kunne implementeres flere category cards ved at lave et nyt objekt i Databasen. Et andet eksempel på et komponent er komponentet solution choice. Dette komponent henter sin information, som eksempelvis billede,  ud fra hvilken robot der er blevet trykket på. Dette fungerer ved at komponenten kigger på Pinia, og når en robot er valgt, sendes informationen til Pinia og herefter videre til komponentet. Dette gør komponentet dynamisk.

**Dynamisk asset referrals S**

Når der arbejdes med compilers, er det god praksis at placere dynamiske billeder i mappen public, da det giver en hurtigere indlæsningstid og caching muligheder [(Vue.js, 2022)](https://www.zotero.org/google-docs/?zaYrQ2). Mappen Public ligger i rodmappen for en Vue.js applikation, og alt indhold fra denne mappe vil blive kopieret til output mappen “dist” når applikationen bygges. Når billeder placeres i Public mappen, kan de indlæses direkte fra serveren uden at gøre brug af Vue.js kompilerings proces, hvilket kan fremskynde indlæsningen af billederne. Derfor er alle projektets dynamiske billeder lagt i Public folderen, mens ikoner og statiske billeder ligger i Assets mappen.

På siden suggestion, kan der ses forslag for robotter. Når der trykkes på en robot, popper et modal op, der har mere information om denne robot. For at lave denne modal dynamisk, er der brugt “template literals”. Med “template literals” er det muligt at indsætte en dynamisk værdi i en string ved at undslippe strengen med syntaxen ${ }. Dette er blevet brugt til at referere til modalens billede, som dynamisk bliver valgt ud fra hvilken robot er valgt. For at dette kan lade sig gøre, skal billederne ligge i Public mappen, som ikke bliver kompileret.

## Udviklingen af platformen:

## ”Spil-bibliotek” (Database):

Som konsekvens af projektets omdrejningspunkt, Data, er der nøje blevet tænkt over hvordan denne data kan fremskaffes samt opbevares. Da RGAPIen kun tillader 100 kald per 2min foruden yderligere aftale med RIOT, er det vigtigt at finde frem til en løsning der kan fremskaffe så meget relevant data som muligt uden at overskrive kvoten. Det er derfor besluttet at alle dataene skal nedgemmes til egen tilgang. Når der snakkes databasestyringssystemer, især SQL og NoSQL databaser, står det klart, at valget mellem de to har væsentlige konsekvenser, især når der håndteres data i den størrelse som projektet planlægger at gøre. Her er snakke om komplekse datatyper som JSON-Arrays med tusindvis af objekter som hver indehold tusindvis af egenskaber. Det er derfor vigtigt at der forstås hvilken databasesystem der bør vælges. Efter grundig overvejelse er der blevet besluttet at en NoSQL database er det bedste valg for projektet. Dette valg er truffet baseret på baggrund af de før opstillede faktorer, og stemmer overens med projektets krav for datastørrelsen og den skalerbarhed.

Hvor SQL fremstår som det ideelle system til transaktion intensivt arbejde med kraftfulde forespørgselsmuligheder, ser det en række udfordringer i skalerbarheden. Skalerbarhed ses at være meget vigtigt for projektet og da NoSQL brilliere på denne front samt tilbyder formidabel fleksibilitet med det dynamiske skema som er fantastisk til at håndtere komplekse datatyper såsom den store volumen af JSON-Arrays projektet forventer at arbejde med. Tilføj hertil at NoSQL er designet til at optimere dataopbevaring og den hastige adgang der til selv når datamængden eskalerer.  
Herudover forenkles datamanipulation krævet for nedgemningen af data, da NoSQL ofte arbejder i JSON stores, så forventningen er at dataen vil kunne nedgemmes i det format den nedhentes i fra RGAPI’en med lidt til ingen manipulation. NoSQL understøtter herudover også strategien om at gemme data lokalt for at minimere afhængigheden af konstante API-kald til RGAPIen og derved undgå at overskride kvoten.

Valget af NoSQL for projektet understreger altså en strategi til databehandling der vægter skalerbarhed, fleksibilitet og hastighed højt, i håb om at projektet skal kunne håndtere dets datakrav effektivt, samtidig med at det åbner for muligheden for dynamisk udskiftelse af datakilder og åbner op for tilpasningsmuligheder som ville være sværere at tilgå i fremtiden med SQL hvis RIOTGAMES besluttede at skifte datastruktur i fremtiden.

## Nunubot:

Mange af de drivende faktorer bag platformens funktionalitet stammer fra script filen ”nunubot.js”. Navnet på filen, Nunubot, er en kultur reference til et champion skin (et alternativt kostume der alterere en champions udseende). Kostumet tager form af en computer der angriber andre ved at ”rulle kode sammen” og kaste ”kodebolde” efter dem. Nunubot.js håndtere alle dataene der nedhentes fra RGAPIen og fungere som en centraliseret metodeliste der kan kaldes i alle de forskellige vuekomponenter. De fleste af disse funktioner gør brug af biblioteket ”axios” til at håndtere api requests til RGAPI’en.

## MatchList:

For at forklare hvordan dataen behandles vil der tages udgangspunkt I to af hovedfunktioner som håndtere spilhistorik logikken og nedgemningen af data til senere analyse. Nedenfor ses spilhistorik logikken:

Funktionen starter med at definere nogle variabler baseret på den nedhentede brugerprofil. Disse variabler tager form af region som navnet antyder indeholder ”server regionen” og id som henvender til riots unikke brugerid såkaldte ”PUUIDs” som er en unik streng på 76 tegn. Herefter håndteres de forskellige spilregioner ved at omdøbe dem til anvendelige navne. Dette gøres for at sikre, at den korrekte streng tilknyttes API kaldet for at oprette forbindelse til den korrekte server. Funktionen laver nu en asynkron forespørgsel til RGAPIen for at hente brugerens spil-historik. Her bruges det tidligere nævnte bibliotek, Axios, til at foretage API forespørgslen. For at eksekvere forespørgslen kombineres statiske strings med de tidligere definerede variabler region og id samt projektets uddelte API nøgle fundet i en separat .env fil i URL'en og sammensat dannes den fulde HTTP query til at eksekvere API-forespørgslen.

A screen shot of a computer code

Description automatically generated

Efter at have nedhentet dataene, itererer funktionen gennem den modtagne spil-historik. For hver kamp, indhenter den yderligere detaljer om kampen og opdaterer en liste af de champions der blev brugt i spillet.

A screen shot of a computer code

Description automatically generated

Efter at have nedgemt spil-historikken I pinia staten ”*matchData*” itereres der gennem hver spiller, *participants*, disse iterationer indeholder et if statement der tjekker hvorvidt profilens navn, den profil som tilhøre spil-historikken, og den returnerede data fra *matchData.info.participants.summonerName* matcher.

A screen shot of a computer

Description automatically generated

I så fald nedgemmes relevante data der bruges til fremvisningen i viewet, *playerOverview,*, som hvilke items der er blevet købt, hvilken champion der er blevet spillet osv., i et objekt

A screen shot of a computer program

Description automatically generated

Efter spil-dataene er blevet arkiveret i *obj* objektet skubbes objektet ind I pinia staten matchinfo som er et array der indeholder de forskellige kampes data.

A screen shot of a computer

Description automatically generated

Som nævnt tidligere bliver denne data anvendt på viewet *PlayerOverview.vue* til at fremvise diverse informationer om spilleren baseret på data fra kampene. Her ses for eksempel Spil-historikken for den givne spiller. Fremvisningen af dette gøres muligt ved at importere Pinia storen og nedarve de påkrævede data som props i *PlayerMatchList.vue*. Herefter udskrives dataen ved brug af v-for loopet som er en indbygget funktion i vue der gør der muligt inline scripte på det specifikke element og herved danne elementet samt alle de nestede elementer baseret på dette loops iterationer.  
  
A screen shot of a computer

Description automatically generated

A screen shot of a computer code

Description automatically generatedA screenshot of a game

Description automatically generated

## Tierlist:

Funktionen *createTierList(gameMode)* er en asynkron funktion der har til formål at danne en såkaldt ”Tierlist”, en liste over den gennemsnitlige præstation af specifikke champions, som har til formål at belyse hvilke champions der klarer sig bedre end andre gennemsnitsligt baseret på den nuværende opdatering. Funktionen for fodret parameteret gameMode til at beslutte hvilke spilleregler den skal sortere dataen efter. Processen indebærer flere aktiviteter som f. eks. Filtreringen af kampe, aggregeringen af champion specifikke data, beregningen af sinple statestikker, og til sidst rangeres listen baseret på disse statestikker og anden kamp data.

A screen shot of a computer code

Description automatically generated

Funktionen starter med at initialisere et tomt object navngivet *mergedChampions* og en “tæller” navngivet *gameCount*. *mergedChampions* hart il formal at fungere som opbevaringssted for de aggregerede data som findes om hver champion, mens *gameCount* har til funktion at holde styr på antallet af kampe med de angivende spilregler. Yderligere bør det nævnes en essentiel del af funktionen findes i en lokalt liggende JSON-fil navngivet *fakeMatchHistory* som indeholder samtlige kamp data fra 80 kampe udspillet af brugeren ”Scissoring Gwen”.

Funktionen itererer gennem JSON-arrayet *fakeMatchHistory* og filtrerer kampe, der matcher den angivne *gameMode*. For hver iteration som matcher *gameMode* øges *gameCount*. Dette gøres for at sikrer det kun er relevante data der betragtes i den følgende eksekvering. Herudover oprettes en ny variabel *team100Participants* som indeholder en filtreret liste over deltagerne fra kampen der var på hold 1. Dernæst skabes endnu en variabel *team200Winners* der indeholder de spillere der var på hold 2 og vandt deres kamp. Herefter ”mappes” *team200Winners* for at transformere den filtrerede liste af spillere til en ny liste der nu indeholder navnet på den champion de spillede da de vandt. Disse metoder og operationer strukturerer dataene på en måde der gør det muligt senere at udførere statistik på hvilke champions der oftere vinder mod andre.

A screen shot of a computer program

Description automatically generated

For hver kamp arkiveres dataen for hver spiller, herunder information så som deres hold, hvilken champion de spillede, og forskellige præstationsdata som kills, deaths, assists, og andre data. Denne data aggregeres for hver champion i det tidligere oprettede array *mergedChampions* for de champions der allerede kan findes i objektarrayet opdateres deres samlede data ved at lægge de nye værdier til de eksisterende, f. eks. Hvis championen ”Shen” har 4 kills en kamp men 3 i en anden kamp vil dataen aggregeres på championens navn i arrayet og kills vil opdateres til 7. Dette skridt tages for senere at kunne opbygge en mere detaljeret statestik over den gennemsnitlige præstation for hver champion.

A computer screen with white text

Description automatically generated

Efter at have samlet data for alle relevante kampe, fortsætter funktionen med at sortere og analysere yderligere. For hver champion, sorteres deres lostToChampions for at identificere, hvilke modstandere de oftest har tabt til. Dette tilføjer et lag af strategisk indsigt, der kan informere spillere om potentielle matchups.

A screen shot of a computer code

Description automatically generated

Til sidst rangeres champions efter deres ”winpercent” og antallet af spillede kampe. Dette skridt er afgørende for at etablere en tier-liste, der afspejler championens effektivitet og popularitet i den givne spiltilstand. Resultatet bliver en sorteret liste over champions, der giver et klart indblik i deres relative styrke baseret på faktuelle spildata.

A screen shot of a computer program

Description automatically generated

Igennem disse processer af dataindsamling, merging og analyse tilbyder funktionen *createTierlist* en dyb forståelse af champions præstationsevner og kan lede spilleren i at træffe informerede beslutninger om, hvilke champions der er bedst at bruge i en given kampsituation. Funktionens brug af både aggregeret statistik og specifikke match-up data gør den til et kraftfuldt værktøj for at forstå og forbedre spilstrategier.

## Markup (Vue)

I VueJS anvendes markup på en måde, der integrerer HTML med VueJS's dynamiske data og komponenter. Markup i VueJS bruger template tagget som et element, hvori der kan defineres HTML-strukturen for et Vue-komponent. Template tagget, kan altså inkludere standard HTML-tags og/eller andre Vue-komponenter, og binde data til disse gennem Vue's operatore som f.eks. ”v-bind” som tillader data at blive dynamisk bundet til attributterne i dine tags. Hvis dynamisk data bindes til en nestet-komponent, et såkaldt child-komponent, skal der oprettes en property list som er en metode for vue til at sende data ned igennem leddene af komponenterne for at fremvise et eksempel på hvordan markup’en struktureres tages der udgangspunk i følgende eksempel fra *PlayerLeftSidebar.vue*:

A screen shot of a computer program

Description automatically generated

Vue komponenten startes med et Template tag. Tagget template er den del, der definerer komponentens HTML-struktur. Alt indenfor <template> vil derfor blive behandlet af Vue og omdannet til det endelige HTML, der fremvises til brugeren.

Inden i template tagget, er der en <div> med klassen contentWrapperLeft, som fungerer som en container for resten af komponenterne.

Inden i denne div container, bliver der refereret til tre Vue-komponenter: *PlayerRanks.vue*, *PlayerChampionStats.vue*, og *PlayerRecentPlayers.vue*. Hver af disse komponenter er designet til at håndtere forskellige dele af spillerens data, såsom deres ranks, spillerens champion statistikker og spillere, spilleren ofte spiller på hold med.

For at kunne fremvise reelle data for brugeren behøver disse komponenter dog visse data. For at videregive disse, anvendes den tidligere nævnte propertylist, forkortet til props i vue. Props er brugerdefinerede attributter, der modtager data fra deres parent-komponent og nedarves et child-komponent. I eksemplet videregives props’ne soloRankStats, flexRankStats, championStats og recentlyPlayedWith til komponenterne ved brug af v-bind: operationen der forkortes med et simpelt kolon ( : ), når man gør brug af :v-bind, indikeres der at værdierne af disse attributter er JavaScript.

A screen shot of a computer program

Description automatically generated

Da props gør brug af reaktiv dynamisk data, betyder dette at når værdierne af disse props ændrer sig i forælderkomponenten (*PlayerLeftsidebar*), vil de også ændre sig i barnekomponenterne(*PlayerChampionStats*), hvilket sikrer reaktiv opdatering af data i brugergrænsefladen. Skulle der forekomme en ændring i dataen ville indholdet af hjemmesiden altså også ændre sig.

## Key NodeJS Pakker:

Axios

Axios kan i vue.js projekter bruges til at håndtere asynkrone datakald. I dette projekt er Axios brugt i Pinia Stores til at hente data fra Firebase Realtime Database. Dette sker ved at Axios-biblioteket sender en GET anmodning til Firebase Realtime Database API-endpoint. Hvis der kigges i koden i Pinia Store kaldt RobotData.js, ses det at når der udføres en asynkron anmodning, bruges await-syntaksen for at vente på at denne anmodning er færdigbehandlet. Herefter gemmes resultatet i DataObj i Storen. På den måde er dataen tilgængelig for hele applikationen

# Evolution:

## Statistik side

## Forside

## Helhedsoplevelse

## Rengøring af kode

# Konklusion:

## Perspektivering: