Realtime Summoner Stats
Tierlists and Pro Tracking

Rapporten er udarbejdet af:
Nicklas Christensen
University College Lillebælt
Marts 2024, 3 Semester, Webudvikling,
hold: WUOE22

Vejleder: Morten Sabro Damgaard

Design Prototype:
https://xd.adobe.com/view/5463d06c-3efa-41e2-b267-3f59c18b94e3-0130/

GitHub:
https://github.com/NickuWasTaken/lolstats

Anbefalede søgeord: ("Scissoring Gwen", "Nicku")

Antal Tegn: 47.826

Abstract:

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. As the game has grown its following so has the number of enthusiasts who crave detailed statistical insights into their gameplay and the broader LoL esports scene. This paper follows a project for a web platform that attempts to bridge this gap. At its core, the project leverages the rich data landscape offered by RIOTS API, to manipulate data and serve it in bite-sized packages to help stimulate its userbases knowledge base through META analysis, player performance metrics, and champion statistics, to deliver a comprehensive analytical platform. A significant decision within the project was choosing between SQL and NoSQL databases, a choice that carries considerable implications for handling large-scale data, particularly complex data types like JSON-Arrays filled with thousands of objects. After careful consideration, NoSQL was chosen for its scalability, flexibility, and the ability to handle complex data structures, aligning with the project's requirements for managing extensive volumes of game data. However, the project does not only focus on data storage and management but also on the user interface and experience, employing tools and technologies that enhance data interaction and presentation. This includes a dedicated method list managed by a script, "nunubot.js", for data retrieval and manipulation, and VueJS for dynamic markup trough databinding's, that resolves in responsive UI constructions. Moreover, the LOLstats project aims to provide insights into player behaviors, preferences, and the competitive scene, offering tools like tier lists and game mode analyses to help players make informed decisions and improve their game strategy. In essence, LOLSTATS aspires to be a holistic solution for the LoL community, offering deep statistical insights that cater to players at all levels, from casual gamers to competitive esports professionals, thereby enriching their gaming experience and performance.

Indholdsfortegnelse

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.

Problemformulering

I takt med League of Legends' voksende popularitet, står spillere over for udfordringen med at navigere i en overflod af spredt information, der er afgørende for at forbedre deres færdigheder og rang. "LOLSTATS" projektet sigter mod at tackle dette problem ved at skabe en centraliseret platform, der aggregerer vigtige data og præsenterer dem gennem en intuitiv brugergrænseflade. Dette rejser spørgsmålet: Hvordan kan man effektivt vise udviklingsmuligheder for LoL-entusiaster på brugergrænsefladen og hvilke teknologier kan optimere håndteringen af store datamængder?

Problemformuleringen:

Hvordan kan man effektivt vise udviklingsmuligheder for LoL-entusiaster på brugergrænsefladen og hvilke teknologier kan optimere håndteringen af store datamængder?

Underspørgsmål:

På hvilken måde kan dataene fra brugergrænsefladen fremvises overskueligt?

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

Afgrænsning

Afspejlet i den følgende rapport vil den opmærksomme læser bemærke at projektet har lagt særligt fokus på spilleroversigten samt tierlisten 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.

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 udviklingen af projektet men også som skabelon til rapporten for at fremme oplevelsen hos læseren.

Persona (Dam and Siang, 2024)
Udformningen af en persona bidrager til at tydeliggøre, hvilken målgruppe ens klient, og dermed projekt, sigter efter at ramme.

Målgruppe analyse (Kursusfabrikken, 2023)
Målgruppeanalysen er brugt til at forstå og definere den specifikke gruppe af mennesker, et platformen er rettet imod. Analysen er begrundet på indsamlet data om målgruppens demografiske træk, interesser, behov, adfærdsmønstre og præferencer.

Prototyping (IDEO, 2012, s. 58):
Prototyping er brugt til at fremstille en design fil udvikleren kan søge til råds hvis der er spørgsmål til udviklingen.

Primær- og Sekundær research, Data indsamling (Svendsen, 2015)
Med primær og sekundær research, er der blevet indsamlet data til projektet. Primær research er indhentet ved interview mens sekundær research er indhentet ved undersøgelse af konkurrenter, samt informationssøgning om det relaterede teknologier.

NoSQL Database model Diagramming (Awan, n.d.)
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 forklarer de forskellige termer der gøres brug af i online fællesskaber omhandlende League of Legends som platformen appellerer 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. Items: Genstande spilleren kan købe for at styrke championens krafter
4. SOLO/DUO: Den primære spilmetode i League of Legends, hvor holdene kæmper om at ødelægge modstandernes Nexus.
5. Lane: De tre hovedstier (Top, Mid, Bot) på Summoner's Rift, hvor kampe finder sted.
6. Jungle: Det område mellem lanerne fyldt med monstre; junglen er ansvaret for at nedbryde monstre og hjælpe lanerne.
7. CS (Creature Score): Antallet af dræbte fjendtlige minions (creeps); det bruges til at måle en spillers evne til at samle guld.
8. Smurfing: Refererer til en spiller som spiller på en alternativ ”account” der er rangeret lavere end hvad spilleren normalt er.
9. Main: Spillerens normale bruger, kan alternativt referere til brugerens højst rangerede ”account” hvis de har flere
10. Buff/Nerf: Positive eller negative effekter, der påvirker en champions evner og-/eller statistikker.
11. Objective: Et kraftfuldt monster, hvis drab giver holdet en permanent fordel.
12. Ultimate: En champions mest kraftfulde evne, ofte låst op på niveau 6.
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 desværre 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 riotgames 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".

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 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.

Delkonklusion:

Igennem discovery-fasen har har LOLSTATS identificeret nødvendige datakilder samt udforsket konkurrenternes styrker og svagheder, for at stille sig i en position hvor der kan skabes en platform der effektivt aggregere og præsentere data for LoL-entusiaster. Ved at kombinere data fra Riot Games' API, Riot Data Dragon, og andre relevante kilder, vil LOLSTATS kunne adressere brugerens behov for dybdegående dataanalyser. Dette samt planer om at aggregere og integrere underholdnings muligheder gennem streaming af professionelle kampe danner grundlaget for at kunne tilbyde en unik værdi, der differentierer sig fra konkurrenterne gennem brugervenlighed, underholdning og integrering af advanceret dataanalyse. Fremadrettet fokuserer projektet på at optimere datahåndtering og fremvisning, hvilket sigter at styrke platformens position som en foretrukket kilde til spilindsigt.

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.(“Understanding the League of Legends Target Audience – Openr,” 2023) 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. (Kursusfabrikken, 2023)

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.

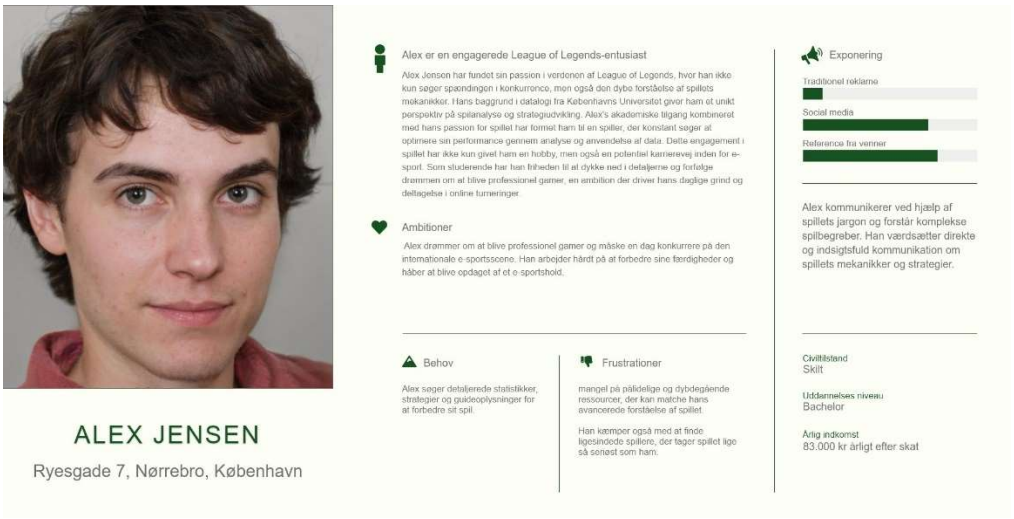

Figure 1 - Persona - Alex Jensen

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.

Figure 2 - Persona - Christian Pedersen

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 giver 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 vedligeholdelsesfordelene 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. (“Vue.js vs. React.js,” 2023)

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 simplificerer en masse udviklingsproblemer der ellers kan forekomme med funktionaliteten der behøver data fra andre side/stadier af webplatformen. Pinia er brugt til at håndtere mange af de API kald der foretages på platformen og distribuere dem herefter ud til de komponenter der skal gøre brug af dem. Ved at importere dataene asynkront fra Pinia-storen fås der også mulighed for at gøre brug af await operatoren der sikrer sig at der er adgang til den påkrævede data før hjemmesidens markup renderes. Mere om brugen af pinia-storen *nunubot.js* kan læses senere i rapporten.

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. (“SQL vs NoSQL Databases,” n.d.)

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 Pinia-storen og 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.

```
async fetchSummonerMatchListById() {
  let region = this.summonerRegion
  let id = this.profileData.puuid
  if (region == 'EUW1') {···
  } else if (region == 'EUN1') {···
  } else if (region == 'OCE1') {···
  } else if (region == 'NA1') {···
  }

  let response = await axios.get(
    'https://' +  region +  `.api.riotgames.com/lol/match/v5
    /matches/by-puuid/` + id + '/ids?1673481600&count=20&api_key=' +
      import.meta.env.VITE_RGAPI
  )
  this.matchHistory = response.data
```

Figure 3 - Kodestykke omhandlende funktionen: FetchSummonerMatchListById

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.

```
let i = 0
while (i < this.matchHistory.length) {
  const currentMatch = this.matchHistory[i]
  await this.fetchMatchDataById(region, currentMatch)
  let p = 0
  var championsInGame = []
  var championsInGameName = {}
```

Figure 4- Kode - while løkke der iterere gennem alle spil

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.

```
while (p < 10) {
  championsInGameName[i] = this.matchData[i].info.participants[p].championName
  await championsInGame.push(championsInGameName[i])
  if (this.matchData[i].info.participants[p].summonerName === this.profileData.
```

Figure 5 - Kode - While løkke der iterere gennem all deltagere

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

```
if (this.matchData[i].info.participants[p].summonerName === this.profileData.name) {
  var obj = {
    summonerName: this.profileData.name,
    team: this.matchData[i].info.participants[p].teamId,
    item0: this.matchData[i].info.participants[p].item0,
    item1: this.matchData[i].info.participants[p].item1,
    ...
  }
```

Figure 6 - Kode - If statement der opretter objekt

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.

```
    p++
  }
  this.matchInfo.matchHistory = i
  await this.matchInfo.push(obj)
  i++
  if (i % 15 == 1) {
```

Figure 7 - Kode - Array.push af objekt

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.

```
import { nunubot } from '@/stores/nunubot'
const lolstats = nunubot()
await lolstats.fetchSummonerByName(lolstats.summonerRegion, lolstats.summonerName)
await lolstats.fetchSummonerLeagueByEncId()
await lolstats.fetchSummonerMatchListById(lolstats.summonerRegion, lolstats.profileData.puuid)
```

Figure 8 - liste over eksekverede funktioner

Figure 10 - Grafik - Match history efter styling og datakald

```
<div class="contentWrapperMid">
  <PlayerMatchList
    v-for="matches in lolstats.matchInfo"
    :match="matches"
    :matchData="lolstats.matchData"
    :key="matches.id"
  />
</div>
```

Figure 9 - Kode - Vue Markup

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.

```
async createTierList(gameMode) {
  let mergedChampions = {}
  let gameCount = 0

  fakeMatchHistory.forEach((game) => {
    if (game.info.gameMode === gameMode) {
      gameCount++
      let team100Participants = game.info.participants.filter((p) => p.teamId === 100)
      let team200Winners = game.info.participants
        .filter((p) => p.teamId === 200 && p.win)
        .map((p) => p.championName)
```

Figure 11 - Kode - Funktionen createTieList()

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.

Figure 12 - Kode - forEach løkke for hver deltager

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.

```
if (!win && team200Winners.length > 0) {
  team200Winners.forEach((winnerChampionName) => {
    if (mergedChampions[championName].lostToChampions[winnerChampionName]) {
      mergedChampions[championName].lostToChampions[winnerChampionName] += 1
    } else {
      mergedChampions[championName].lostToChampions[winnerChampionName] = 1
    }
  })
}
```

Figure 13 - Kode - If statement der checker kamp vindere

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.

```
// Sortere lostToChampions for hver champion
Object.values(mergedChampions).forEach((champion) => {
  champion.lostToChampionsCount = Object.entries(champion.lostToChampions)
    .map(([name, count]) => ({ name, count }))
    .sort((a, b) => b.count - a.count)
})

const mergedArray = Object.values(mergedChampions)
```

Figure 14 - Kode - forEach statement der sortere counters

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.

```
  // Sorter mergedArray efter winrate, derefter spil mængde
  mergedArray.sort((a, b) => {
    const winPercentageA = (a.wins / a.gamesPlayed) * 100
    const winPercentageB = (b.wins / b.gamesPlayed) * 100

    if (winPercentageB === winPercentageA) {
      return b.gamesPlayed - a.gamesPlayed
    }
    return winPercentageB - winPercentageA
  })
  this.tierlistData = mergedArray
  this.tierlistData.allGames = gameCount
}
```

Figure 15 - Kode - Array.sort() metode der retunere det slutlige sorterede array

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*:

```
<template>
  <div class="contentWrapperLeft">
    <PlayerRanks
    :soloRankStats="props.soloRankStats"
    :flexRankStats="props.flexRankStats"
     />

    <div>
      <PlayerChampionStats
      :champions="props.championStats"
      />
    </div>

    <div><PlayerRecentPlayers :recentlyPlayedWith="props.recentlyPlayedWith" /></div>
  </div>
</template>
```

Figure 16 - Kode - VueJS komponenter

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 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.

```
<script setup>
const props = defineProps({
  champions: {}
})
</script>

<template>
<div class="leftTab leftTab--champion" v-for="champion in props.champions">
  <img
    :src="`src/assets/GameAssets/champion/${champion.championName}.png`"
    alt=""
    class="championThumb"
  />
```

Figure 17 - Kode - VueJS props declaration

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

Delkonklusion:

Denne del af rapporten omhandlede effektiv udnyttelse af data, med særlig fokus på udfordringen ved den begrænsede RGAPI fra RIOT. For at navigere inden for disse restriktioner, prioriteres lokalisering af data gennem nedgemning, hvilket kræver omhyggelig overvejelse af databasestyringssystemer. Projektets datastørrelse og kompleksitet, herunder håndtering af komplekse JSON-Arrays, har medført en foretrukken anvendelse af NoSQL databaser. Denne beslutning understøtter projektets krav til skalerbarhed, fleksibilitet, og hastighed ved databehandling. NoSQL's dynamiske skemaer og evne til at opbevare data i JSON format muliggør effektiv datamanipulation og tilpasningsdygtighed til fremtidige datastrukturændringer fra RIOTGAMES.

Udover databehandlingen gik denne del også i dybden med nogle af statemanagerens funktioner som spilhistoriklogik og tier-lists gennem nunubot.js, der udnytter node pakken axios til at simplificere API-kald, og Pinia for state-management. Disse funktioner faciliteter indsamling, behandling, og præsentation af data vedrørende spillers præstationer og champion statistikker.

Projektet integration af vueJS tillader en dynamisk og reaktiv brugerflade hvor data kan bindes til markup elementer for at understøtte en mere effektiv datastrøm mellem komponenter.

Key NodeJS Pakker:

Axios

Axios er i projektet brugt til at håndtere og simplificere asynkrone datakald. I dette projekt er Axios brugt i Pinia Storen *nunubot.js* til at hente data fra RGAPI'en. Dette sker ved at Axios-biblioteket sender en GET-anmodning til et RGAPI-endpoint. Ser man nærmere på koden i Pinia Storen kaldt *nunubot.js*, ses det at når der udføres en asynkron anmodning, bruges await-syntaksen for at afvente at anmodningerne er færdigbehandlede. Herefter behandles dataene og oftest gemmes resultatet som en pinia-state. På den måde er dataen tilgængelig for hele applikationen.

Evolution:

Forside

Når siden åbnes, vil der straks ligges mærke til at platform ikke fremviser dynamisk data om de professionelle ligaer som ellers er planlagt og designet i prototypen. Den primære årsag af denne afvigelse ligger i begrænsninger ved adgangen til denne specifikke data gennem RGAPI, som ikke tilbyder direkte eller pålidelige datastrømme for professionel ligaaktivitet. Alternativt er der senere i forløbet blevet fundet en Python-pakke gennem web-forrummet Reddit udviklet af Leo Lo, der potentielt kan anvendes i fremtidige iterationer af hjemmesiden for at inkorporere denne data og dermed berige brugeroplevelsen med aktuelle og relevante informationer om den professionelle liga. Dog er designet øjeblikkeligt statisk.

Den Professionelle scene

Som beskrevet ovenfor har det været umuligt at ekstrapolere data fra RGAPIen omkring det professionelle miljø. Dette har ført til at et manglende af aspekt af hjemmesiden. Originalt planlagdes udviklingen, af integrerede data der ville formidle information omkring kampe, afspillede blandt de forskellige professionelle ligaer. Derfor inkludere denne iteration af platformen hverken den dedikerede videoside eller rangeringslisten over professionelle hold. Andre elementer involverende det professionelle miljø, får dets data serveret statisk og udviklet udelukkende som placeholders for at give et fuldhedsudtryk af platformen.

Filtrering

Grundet visionen bag platformen dataintensive og analytiske tilgang ses der mange tiltænkte filtrerings former tilhørende de forskellige sider på platformen. Ikke alle er nået at blive implementeret. På tierlist, kan der filtreres på roller, kamp type og combat stats, DUO statestikker har dog ført til besvær da der under udarbejdelsen af projektet ikke har kunne ses hvordan disse statestikker føres. Der tages dog ansvar for at dette er en mangel for kun

for denne platform da konkurrenterne tilbyder dog disse data så det må være set muligt at bearbejde nogle af dataene til at få pålideligt information om specifikt hvem der har ”duo queue’d”. Derfor ses udarbejdelsen af denne funktion som en udviklingsmulighed i fremtidige iterationer sammen med filtreringen for professionelle.

Champion Side

Sidst blandt det planlagte design fra prototypen ses ”champion” siden der var ment at udlove statistik og data vedrørende alle de sekundere funktioners indflydelse på en champions præstationsevne. Denne side var ment at inkludere specificerede informationer om optimale items der bør købes, hvilke ’ability upgrades’ der bør prioriteres, hvilke runer man bør gøre brug af samt meget mere. Det har dog ikke været muligt at udvikle denne side inden for tidsrammen af forløbet, men ville ubetvilvtsomt være det det næste skridt i udviklingen, for at fuldende platformen databehandlings-aspektet samt statistikker. Da denne side ikke er blevet udviklet, medfølges det selvfølgelig at champions ikke kan søges i søgefeltet.

Yderligere udvikling

Ses der mod horisonten længere end den fuldtudviklet platform med alle det forud designede funktionaliteter menes LOLSTATS som platform at kunne udvides på disse måder

Udvidelse af spildækningen: Flere af konkurrenterne følger en trend der tager udgangspunkt i at inkorporere en bredere vifte af populære spil og e-sportstitler. LOLSTATS mener at kunne følge denne trend og herved udvide sit potentiale for at tiltrække en diversificeret brugerskare med forskellige interesser inden for e-sportsverdenen.

Integration af sociale funktioner: En element mange af konkurrenterne gør brug af er inkorporeringen af gamification, der udloves specifikke tags, eller emblemer til brugere der opnår milepæle gennem deres ingame præstationer. Samtidig fandtes dette at være et painpoint da hos udspurgte brugere da der ikke var nogen måde at få et overblik over disse emblemer. LOLSTATS mener at kunne dette bedre ved at udvikle of fremstille et direktiv der tillader spillere at se alle udfordringer og milepæle. Hertil kunne også tilføjes vedvarende profiler for platformen der kan dele deres emblem og måske i fremtiden inddrage spillere til at udfordre hinanden. Ved at tillade brugere at oprette profiler, dele deres statistikker og deltage i diskussioner, fremmes et fællesskab omkring LOLSTATS. Dette skaber et miljø, hvor brugere ikke kun kommer for data, men også for samvær og udveksling af viden. (*Social Media and Gamification (Video)*, n.d.)

Udvikling af en mobilapp: I en tid, hvor mobilenheder dominerer, vil udviklingen af en dedikeret mobilapp gøre platformen mere tilgængelig for brugere på farten og/eller in-game. En app kan tilbyde opdateringer og gør det lettere for brugere at forbruge indhold.

Etablering af Premium tjenester: Ved at tilbyde abonnementsbaserede premiumtjenester kan LOLSTATS generere yderligere indtægter, samtidig med at der leveres værdi til brugere gennem avancerede analyser, personlige anbefalinger og eksklusivt indhold.

Gennem disse initiativer kan platformen ikke kun cementere sin position som en førende informationskilde inden for gaming og e-sports, men også skabe en rigere, mere engagerende platform for et voksende fællesskab.

Konklusion:

Gennem udarbejdelsen af dette projekt er der blevet analyseret på LOLSTATS som en platform samt lavet grundig research på statistik-sfæren, LOLSTATS mulige konkurrenter, platformens målgruppe og muligheden for at indtage en plads på hvad umiddelbart virker til at være et redocean market. Denne proces har lagt grundlag for den videre produktudvikling der er blevet udtænkt, samt givet bedre forståelse og empati for målgruppens behov og frustrationer. Der var ikke megen ide generering involveret specifikt i dette forløb da design prototypen allerede var udviklet og leveret, men der er alligevel blevet gjort en del tanker om hvordan data kunne struktureres og bearbejdes på forhånd for udviklingen af platform men selvfølgeligvis også under udviklingen. Langt største delen af projektet har fundet tid i udviklingsfasen (eksperimentation) som det oftest er tilfældet når der gælder større webprojekter. Projektets tog hånd om et problem der udfoldede sig I data repræsention, manipulation og optimeringsmuligheder og selvom platformen ikke nåede den fulde udvikling, menes der blandt den udviklede del af platformen og design prototypen at være nået et punkt hvortil kan ved kombination få et helheds indblik i hvordan siden vil tage form. Ved brugen af de forskellige teknologier beskrevet gennem rapporten og det letvægtige framework vueJS, er platformen blevet udviklet i en form der er tilegnet håndteringen af store datamængder. Samtidig formidles disse datamængder på en grafisk overskuelig måde der er brugernem og giver brugeren masserer at mulighed for at filtrere of sortere data der giver brugen adgang til præcist den information de søger is deres rejse mod forbedring. I den nære fremtid er håbet at kunne implementere videre datahåndtering, specielt omkring den planlagte e-sports aggregation der originalt var planlagt for platformen og herved opfylde et unikt behov der ikke opfyldes hos konkurrenterne og derved skabe værdi der kan tiltrække og vedligeholde en brugerbase til platformen.

Literaturliste:

Awan, T., n.d. How to Draw NoSQL Data Model Diagram? [WWW Document]. TecHighness. URL https://www.techighness.com/post/how-to-draw-no-sql-data-model-diagram/ (accessed 1.2.24).

Dam, R.F., Siang, T.Y., 2024. Personas – A Simple Introduction [WWW Document]. Interact. Des. Found. URL https://www.interaction-design.org/literature/article/personas-why-and-how-you-should-use-them (accessed 3.8.24).

Kursusfabrikken, 2023. Målgruppeanalyse - Hvordan bliver du skarp på din målgruppe? Kursusfabrikken. URL https://www.kursusfabrikken.dk/maalgruppeanalyse/ (accessed 1.2.24).

Social Media and Gamification (Video), n.d.

SQL vs NoSQL Databases: Key Differences and Practical Insights [WWW Document], n.d. URL https://www.datacamp.com/blog/sql-vs-nosql-databases (accessed 3.7.24).

Svendsen, S., 2015. Primær og sekundær data. Det Gyldne Overblik - Akad. Opgaveskrivning. URL http://detgyldneoverblik.dk/primaer-sekundaer-data/ (accessed 3.8.24).

Understanding the League of Legends Target Audience – Openr, 2023. URL https://openr.co/understanding-the-league-of-legends-target-audience/ (accessed 3.8.24).

Vue.js vs. React.js: A Comprehensive Comparison, 2023. URL https://www.webcoding.dev/vue-js/vuejs-vs-reactjs-a-comprehensive-comparison/ (accessed 3.7.24).

Saffer, D. 2010. Design for Interaction. 2. udgave. Berkeley, CA: New Riders

IDEO 2012. “Design thinking for educators” 2. udgave, Palo Alto, CA, IDEO LLC,

Bilag:

Bilag 1:

ALEX JENSEN

Ryesgade 7, Nørrebro, København

Alex er en engagerede League of Legends-entusiast

Alex Jensen har fundet sin passion i verdenen af League of Legends, hvor han ikke kun søger spændingen i konkurrence, men også den dybe forståelse af spillets mekanikker. Hans baggrund i datalogi fra Københavns Universitet giver ham et unikt perspektiv på spilanalyse og strategiudvikling. Alex's akademiske tilgang kombineret med hans passion for spillet har formet ham til en spiller, der konstant søger at optimere sin performance gennem analyse og anvendelse af data. Dette engagement i spillet har ikke kun givet ham en hobby, men også en potentiel karrierevej inden for e-sport. Som studerende har han friheden til at dykke ned i detaljerne og forfølge drømmen om at blive professionel gamer, en ambition der driver hans daglige grind og deltagelse i online turneringer.

Ambitioner

Alex drømmer om at blive professionel gamer og måske en dag konkurrere på den internationale e-sportsscene. Han arbejder hårdt på at forbedre sine færdigheder og håber at blive opdaget af et e-sportshold.

Behov

Alex søger detaljerede statistikker, strategier og guideoplysninger for at forbedre sit spil.

Frustrationer

mangel på pålidelige og dybdegående ressourcer, der kan matche hans avancerede forståelse af spillet.

Han kæmper også med at finde ligesindede spillere, der tager spillet lige så seriøst som ham.

Exponering

Traditionel reklame

Social media

Reference fra venner

Alex kommunikerer ved hjælp af spillets jargon og forstår komplekse spilbegreber. Han værdsætter direkte og indsigtsfuld kommunikation om spillets mekanikker og strategier.

Civiltilstand
Skilt

Uddannelses niveau
Bachelor

Årlig indkomst
83.000 kr årligt efter skat

Bilag 2

CHRISTIAN PEDERSEN

Trøjborgvej 32, Århus

Christian Pedersen en casual League of Legends-spiller

Christian Pedersen repræsenterer det "casual" segment af League of Legends-spillere. Arbejdslivet som projektleder for en IT-startup i Aarhus fylder meget i hans hverdag, hvilket gør gaming til en værdsat afkobling og en måde at opretholde sociale forbindelser. Charlie nyder at fordybe sig i spillet som en måde at slappe af på efter lange arbejdsdage og værdsætter især det fællesskab og de venskaber, spillet har bragt ham. Selvom han ikke har ambitioner om at blive professionel, er han ivrig efter at forbedre sin spiloplevelse uden at det skal føles som endnu et projekt. Charlie balancerer sin hobby med et aktivt personligt liv og ser værdi i spillets evne til at tilbyde hurtig underholdning og en flugt fra hverdagens stress.

Ambitioner

Christians reelle ambitioner ligger uden for spillet og ser kun spillet som underholdning herfor er det vigtigt for Christian at modtage letfordøjelige guides, underholdende spilfakta, og tips, der forbedrer hans spiloplevelse uden at kræve for meget tid eller fordybelse. Casual gameplay tips og sociale aspekter af spillet, som hvordan man spiller med venner, er særligt værdifulde for ham.

Behov

Interesseret i generel vejledning og tips til at nyde spillet mere og forstå dets grundlæggende.

Frustrationer

Christian bliver frustreret over for komplekst indhold, der føles irrelevant for hans casual spiloplevelse. Han kæmper også med at holde trit med spillets skiftende meta, som kræver tid til at lære.

Exponering

Traditionel reklame

Social media

Reference fra venner

Charlies sprogbrug er mere afslappet og mindre teknisk. Han foretrækker simpel og letforståelig information, der kan hjælpe ham med at nyde spillet bedre uden at føle sig overvældet.

Civiltilstand
I et forhold

Uddannelses niveau
Kandidat

Årlig indkomst
461.000 kr årligt efter skat