Bacheloroppgave

IE303612 Bacheloroppgave Data

System for håndtering av spillerlogistikk for Aalesunds fotballklubb

10034, 10036, 10037

Totalt antall sider inkludert forsiden: 58 (87)

Ålesund, Innleveringsdato 20.05.19

Obligatorisk egenerklæring/gruppeerklæring

Den enkelte student er selv ansvarlig for å sette seg inn i hva som er lovlige hjelpemidler, retningslinjer for bruk av disse og regler om kildebruk. Erklæringen skal bevisstgjøre studentene på deres ansvar og hvilke konsekvenser fusk kan medføre. Manglende erklæring fritar ikke studentene fra sitt ansvar.

Du/dere fyller ut erklæringen ved å klikke i ruten til høyre for den enkelte del 1-6:		
1.	**Jeg/vi erklærer herved at min/vår besvarelse er mitt/vårt eget arbeid, og at jeg/vi ikke har brukt andre kilder eller har mottatt annen hjelp enn det som er nevnt i besvarelsen.**	☒
2.	**Jeg/vi erklærer videre at denne besvarelsen:** • ikke har vært brukt til annen eksamen ved annen avdeling/universitet/høgskole innenlands eller utenlands. • ikke refererer til andres arbeid uten at det er oppgitt. • ikke refererer til eget tidligere arbeid uten at det er oppgitt. • har alle referansene oppgitt i litteraturlisten. • ikke er en kopi, duplikat eller avskrift av andres arbeid eller besvarelse.	☒
3.	**Jeg/vi er kjent med at brudd på ovennevnte er å betrakte som fusk og kan medføre annullering av eksamen og utestengelse fra universiteter og høgskoler i Norge, jf. Universitets- og høgskoleloven §§4-7 og 4-8 og Forskrift om eksamen §§14 og 15.**	☒
4.	**Jeg/vi er kjent med at alle innleverte oppgaver kan bli plagiatkontrollert i Ephorus, se Retningslinjer for elektronisk innlevering og publisering av studiepoenggivende studentoppgaver**	☒
5.	**Jeg/vi er kjent med at høgskolen vil behandle alle saker hvor det forligger mistanke om fusk etter høgskolens studieforskrift §31**	☒
6.	**Jeg/vi har satt oss inn i regler og retningslinjer i bruk av kilder og referanser på biblioteket sine nettsider**	☒

Publiseringsavtale

Studiepoeng: 20

Veileder: Di Wu, Anniken Karlsen

Fullmakt til elektronisk publisering av oppgaven

Forfatter(ne) har opphavsrett til oppgaven. Det betyr blant annet enerett til å gjøre verket tilgjengelig for allmennheten (Åndsverkloven §2).
Alle oppgaver som fyller kriteriene vil bli registrert og publisert i Brage HiM med forfatter(ne)s godkjennelse.
Oppgaver som er unntatt offentlighet eller båndlagt vil ikke bli publisert.

Jeg/vi gir herved NTNU i Ålesund en vederlagsfri rett til å gjøre oppgaven tilgjengelig for elektronisk publisering:	☒**ja**	☐**nei**
Er oppgaven båndlagt (konfidensiell)? (Båndleggingsavtale må fylles ut) - Hvis ja: **Kan oppgaven publiseres når båndleggingsperioden er over?**	☐**ja** ☐**ja**	☒**nei** ☐**nei**
Er oppgaven unntatt offentlighet? (inneholder taushetsbelagt informasjon. Jfr. Offl. §13/Fvl. §13)	☐**ja**	☒**nei**

Dato: 20.05.19

BACHELOROPPGAVE - RAPPORT

NTNU
Kunnskap for en bedre verden

TITTEL:
System for håndtering av spillerlogistikk for Aalesunds fotballklubb

KANDIDATNUMMER(E):
10034, 10036, 10037

DATO:	EMNEKODE:	EMNE:	DOKUMENT TILGANG:
20.05.19	IE303612	Bacheloroppgave	

STUDIUM:	ANT SIDER/VEDLEGG:	BIBL. NR:
Bachelorstudium, Data	58 / 4	

VEILEDER(E):
Di Wu, Anniken Karlsen

SAMMENDRAG:

Denne rapporten beskriver grunnlaget for, gjennomføringen-, og resultatet av en bacheloroppgave gjennomført ved NTNU i Ålesund. Oppgaven er gitt av Bjørn Erik Melland, sportslig leder for Aalesunds FK. Oppgaven var å forenkle arbeidet til sportslig leder ved å lage et system for behandling av spillerdata. Dette er informasjon sportslig leder benytter til vurdering og rekrutering av spillere. Informasjonen kan også innhentes av speidere som samarbeider med klubben.

Løsningen ble å lage en web-basert applikasjon med brukergrensesnitt basert på HTML, CSS og JavaScript. Applikasjonen skulle være tilgjengelig for sportslig leder og speidere, slik at de kunne legge til informasjon om spillere i en database, og mulighet for å redigere data på spillere allerede i databasen. Forbehandlingen av brukergrensesnittet ble gjort gjennom Python med en MySQL-database i bakgrunnen. Applikasjonen skulle kun tillate autorisert innlogging, ha flere brukere, roller til brukere og mulighet til å søke gjennom databasen etter spillere.

Prosjektet brukte en plandreven utviklingsmetode som tok utgangspunkt i kjernefunksjoner som ble oppgitt av oppdragsgiver ved første møte og ble utvidet i løpet av prosjektets gang.

Sluttresultatet er et fungerende produkt som gjør alt som var krevd av det. I tillegg gir den muligheten for å legge til interessante spillere i en favorittgruppe og vise en visuell representasjon av et lag bestående av favoritter. Det er fremdeles noen funksjoner som ikke er optimalisert og noen som ikke er fullstendig ferdigutviklet som bør arbeides videre med, men som er sett på som mindre viktige for prosjektets slutt.

Denne oppgaven er en eksamensbesvarelse utført av studenter ved NTNU i Ålesund.

Postadresse
NRNU i Ålesund
N-6025 Ålesund
Norway

Besøksadresse
Larsgårdsvegen 2
Internett
www.hials.no

Telefon
70 16 12 00
Epostadresse
postmottak@hials.no

Telefax
70 16 13 00

Bankkonto
7694 05 00636
Foretaksregisteret
NO 971 572 140

FORORD

Rapporten har som hensikt å dokumentere gjennomføringen av, hensikten bak, og resultatet av bacheloroppgaven gitt av Aalesunds fotballklubb (AaFK) ved sportslig leder Bjørn Erik Melland.

AaFK er et norsk fotballag som for øyeblikket spiller i Obosligaen. Klubben har en lang og innholdsrik historie som strekker seg tilbake til 1914 og ble blant annet kåret norgesmestere i 2009 og 2011.

Håndtering av spillerlogistikk har vært en reel utfordring hos AaFK. Med en person som ansvarlig for håndtering av all informasjonen; lagret på sin datamaskin, fordelt på forskjellige filer og mapper. Melland har selv lagret spillerdata i et filsystem, men dette systemet vil fremstå som svært rotete for neste person i Mellands stilling. Etterfølger i stillingen som sportslig leder vil trolig måtte danne et nytt system. Dermed vil systemet vi utvikler bistå med både effekt for nåværende ledelse, men også forenkle informasjonsflyten ved innføring av en ny administrasjon.

Melland har over lengre tid hatt fokus på å forbedre spillerlogistikken til AaFK slik at de kan få en fordel ovenfor sine konkurrenter. Med denne oppgaven håper vi at vi har lagt grunnmuren for et system som vil hjelpe Melland og AaFK med det.

Målet har vært å utvikle et digitalt logistikksystem for spillerhåndtering. Løsningen skulle vektlegge spillerrekrutering og speiding. Den skulle gjøre det mulig å lagre spillerprofiler på ett strukturert vis, med mulighet for å lagre og endre spillerprofiler. Løsningen skulle også ha en måte å søke gjennom spillerne i systemet.

Vi ønsker å takke Aalesunds FK og sportslig leder Bjørn Erik Melland for muligheten til å ta del i et spennende prosjekt i et fagfelt med sterk konkurranse og som er i stadig utvikling.

Vi ønsker også å takke Di Wu ved NTNU i Ålesund som har vært veileder for oppgaven og har holdt tett oppfølging gjennom hele prosessen.

En spesiell takk går til Anniken Karlsen som har vært assisterende veileder. Med sin kompetanse og erfaring innen systemutvikling og som veileder, har hun vært til stor hjelp i kritiske faser av prosjektet.

INNHOLD

SAMMENDRAG

Denne rapporten beskriver grunnlaget for, gjennomføringen-, og resultatet av en bacheloroppgave gjennomført ved NTNU i Ålesund. Oppgaven er gitt av Bjørn Erik Melland, sportslig leder for Aalesunds FK. Oppgaven var å forenkle arbeidet til sportslig leder ved å lage et system for behandling av spillerdata. Dette er informasjon sportslig leder benytter til vurdering og rekrutering av spillere. Informasjonen kan også innhentes av speidere som samarbeider med klubben.

Løsningen ble å lage en web-basert applikasjon med brukergrensesnitt basert på HTML, CSS og JavaScript. Applikasjonen skulle være tilgjengelig for sportslig leder og speidere, slik at de kunne legge til informasjon om spillere i en database, og mulighet for å redigere data på spillere allerede i databasen. Forbehandlingen av brukergrensesnittet ble gjort gjennom Python med en MySQL-database i bakgrunnen. Applikasjonen skulle kun tillate autorisert innlogging, ha flere brukere, roller til brukere og mulighet til å søke gjennom databasen etter spillere.

Prosjektet brukte en plandreven utviklingsmetode som tok utgangspunkt i kjernefunksjoner som ble oppgitt av oppdragsgiver ved første møte og ble utvidet i løpet av prosjektets gang.

Sluttresultatet er et fungerende produkt som gjør alt som var krevd av det. I tillegg gir den muligheten for å legge til interessante spillere i en favorittgruppe og vise en visuell representasjon av et lag bestående av favoritter. Det er fremdeles noen funksjoner som ikke er optimalisert og noen som ikke er fullstendig ferdigutviklet som bør arbeides videre med, men som er sett på som mindre viktige for prosjektets slutt.

TERMINOLOGI

Begreper

Tag	En tag, i sammenheng HTML og andre markeringsspråk, er et element som blir satt inn i dokumentet eller en fil for å endre utseende eller utfører en handling (Hope, 11).
Integer	Ofte forkortet int eller INT, er en fundamental variabel type bygd inn i kompileringen og brukes for å definere numeriske variabler og holder på hele tall (Bolton, 2019).
String	En datatype brukt i programmering, som en integer (...), men er brukt for å representere tekst istedenfor nummer (Christensson, 2006).
Entitet	«En entitet kan beskrives ved sine attributter. I analysen forut for konstruksjonen av en database søker en etter entiteter og deres innbyrdes sammenhenger for den virkelighet databasen skal representere» (Bratbergsengen, 2018).
Attributt	«En attributt er navn til en verdi eller flere verdier i en entitet eller post i en database. I relasjonsmodellen er attributtnavnet kolonneoverskriften i en tabell» (Bratbergsengen, 2017).
JSON	Et lettvekts data utvekslings format. (...)JSON er bygd på to strukturer

	- en samling av navn/verdi par. (...) dette er realisert som objekter (...) - en ordnet liste av verdier. (...) dette er realisert som en matrise, vektor, liste eller sekvens (Crockford, u.d.).
Primærnøkkel	En nøkkel i en relasjonsdatabase som er unik for hvert element. Det er en unik identifikasjon (...) en relasjonsdatabase må alltid ha en, og kun en primærnøkkel (Rouse, 2005).
Fremmednøkkel	En fremmednøkkel er i en annen tabell en det den tilhørende primærnøkkelen som hører til (Rouse, 2017).
Varchar	En datatype som kan holde på all type data: numerisk, karakterer, mellomrom og punktum (Technopedia, u.d.).
UML	Et standardisert modelleringsspråk bestående av integrerte sett av diagrammer, utviklet for å bistå system og programvare utviklere med spesifisering, visualisering, oppbygging og dokumentering av programvare systemer (Visual Paradigm, u.d.).
GET	Brukes for å forespørre data fra en spesifikk resurs. GET er en av de vanligste http metodene (w3schools, u.d.)
POST	Brukes for å sende data til serveren for å lage eller oppdatere resurser (w3schools, u.d.)

Symboler

$	Brukes for å initiere AJAX fra JavaScript
()	brukes for å identifisere en funksjon og kan være tom eller inneholde det variabel funksjonen trenger.
/	brukes for å identifisere veien til en spesifikk resurs

Forkortelser

UML	Unified Modeling Language
HTML	Hyper Text Markup Language
CSS	Cascading Style Sheet
JS	JavaScript
SQL	Structured Query Language
AaFK	Aalesunds Fotballklubb
UP	Unified Process
ISO	International Organization for Standardization
JSON	JavaScript Objekt Notasjon
Varchar	Variable Character Field

1 INNLEDNING

Bacheloroppgaven er utført av Espen Nordfjellmark Bårdevik, Fredrik Halsebakke Kvalheim og Storm Østberg med hensikt å utvikle et digitalt logistikksystem ved NTNU i Ålesund våren 2019 etter krav formulert av Aalesunds Fotballklubb:
Oppgaven er motivert av sin relevans til datastudiet vårt hvor databaser, systemutvikling og nettverksutvikling har vært kjernefokus i utdanningen. Oppgaven har gitt oss muligheten til å sette sammen ulike kompetanseområder fra utdanningen til en helhet. Den har også gitt oss muligheten til å jobbe med en reell oppdragsgiver med de utfordringer det innebærer.

Vår oppgave er gitt av AaFK's sportslige leder Bjørn Erik Melland. Han ville ha en måte for AaFK å samle all spillerdata i et system. Slik som praksis i fotballklubben har vært tidligere, ligger mye informasjon relatert til spillerstatistikk spred på forskjellige filer, i forskjellige mapper, på en bestemt datamaskin. Dette medfører selvsagt at når sportslig leder skal skaffe oversikt så bruker han ekstra tid på å hente og sammenstille informasjon mer eller mindre manuelt. Han er også avhengig av den ene maskinen. En annen utfordring er selvsagt kunnskapsoverføring mellom forskjellige brukere av et manuelt system.

Gjennom oppgaven har vi, i samspill med oppdragsgiver, lagt vekt på at systemet skal være intuitivt og enkelt å bruke. Oppdragsgiver har vært klar på at systemets primæroppgave er å forenkle sportslig leders jobb per dags dato. Systemet skulle kunne gjøre følgende:

- Opprette en spiller i en type database med mulighet for å endre informasjonen om spillere.
- Samle all relevant informasjon om en spiller til en oversiktlig profil.
- Lage en liste over spillere i databasen, og mulighet for å søke eller filtrere listen.
- Lage et «skygge-lag», som presenterer mulige spillere i designerte roller.
- Mulighet til å favorisere spillere ut fra hva som er mest relevant for tidspunktet.
- Skrive ut et prospekt av en spiller for å vise til relevant personell.

Oppdragsgiver informerte om at mye av informasjonen om internasjonale spillere blir hentet inn av speidere lokalisert i utlandet. For å forenkle oppdragsgivers jobb, ble det bestemt at det å lage en web-applikasjon ville være ønskelig. På denne måten ville speidere ha mulighet til å legge til spillere med relevant informasjon direkte til databasen i stedet for å gå gjennom prosessen med å sende informasjonen til oppdragsgiver for at oppdragsgiver skal legge det inn manuelt selv.

Melland selv har lagt vekt på at håndtering av spillerlogistikk er en viktig faktor for klubbens framtidige suksess (Stenerud, 2018). AaFK har også vist dette ved å ta grep blant annet ved ansettelse av en konsulent (Mudrinic, 2018) for å ta deler av ansvaret for spillerlogistikk. Spillerlogistikk er tydelig viktig for klubben og prosjektet vårt kan derfor være til stor hjelp i et kritisk område.

Gruppens mål er å lage et logistikksystem for å håndtere den mengden informasjon som blir innhentet ved speiding etter spillere, og vedlikeholde fotballfaglig informasjon om spillerstallen til et fotballag som AaFK. Det web-baserte grensesnittet skal være intuitivt, visuelt estetisk og gløde AaFK.

Dersom sluttproduktet oppfyller disse kravene, vil det kunne motivere klubben til å bruke, investere i, vedlikeholde og videreutvikle systemet for å videre hjelpe klubben til nye høyder.

Problemstilling:
Hvordan kan vi lage et moderne system som AaFK er villig til å bruke fremfor sin nåværende løsning?

Hvordan gjøre systemet brukervennlig og samtidig få det til å være visuelt estetisk?
Hvordan gjøre systemet sikkert i forhold til dagens standard for web-baserte applikasjoner?

Hvordan gjøre det slik at informasjonen som kommer fra forskjellige personer verden over blir riktig, både at de legger inn riktig informasjon og at informasjonen blir riktig fremvist?

2 TEORETISK GRUNNLAG

2.1 Menneske-maskin interaksjon

Når en skal utforme et grafisk brukergrensesnitt er det flere ting å ta hensyn til. Dette er noe som har blitt utforsket og testet, og det er utformet flere prinsipper som vi kan ta i bruk. Dette delkapittelet vil nevne noen kjente prinsipper som ble brukt for dette prosjektet.

2.1.1 Ben Shneiderman's "Åtte Gylne Regler for Grensesnittdesign"

Shneiderman (University of Maryland, u.d.) er en anerkjent universitetsprofessor innen datavitenskap og grunnlegger av Human-Computer Interaction Laboratory hos University of Maryland. Han publiserte i 1986 første versjon av «Designing the User Interface: Strategies for Effective Human-Computer Interaction» (Shneiderman, 1986), der har introduserte sin liste over «Åtte Gylne Regler for Grensesnittdesign» (Shneiderman, 2016):

1. **Streb etter konsistens (Strive for consistency)**.

   For like situasjoner burde lik håndtering være påkrevd; identisk terminologi burde bli brukt for meldingsbokser, menyer, og hjelpeskjermer.

2. **Gå for universell brukervennlighet (Seek universal usability).**

   Anerkjenn at det finnes ulike brukere og design for formbarhet, tilrettelegg for endring av innhold.

3. **Gi informativ tilbakemelding (Offer informative feedback).**

   Hver brukerhandling bør gi en tilbakemelding fra systemet. For små og hyppige handlinger kan tilbakemeldingen være beskjedne, mens for større og sjeldnere handlinger bør tilbakemeldingen være mer tydelig.

4. **Handlinger bør ha en tydelig slutt (Design dialogs to yield closure).**

   En rekke handlinger skal være organisert i grupper med en begynnelse, en midtdel og en slutt. Informativ tilbakemelding etter utført handlinger, gir brukeren tilfredstillelse, følelsen av å ha oppnådd noe og indikasjon om å begynne på neste handling.

5. **Forhindre feil (Prevent errors).**

   Design grensesnittet så godt som mulig for ikke å tillate brukere å gjøre noen alvorlige feil. For eksempel å deaktivere meny elementer som ikke er relevante, eller å ikke tillate bruker å skrive bokstaver i felt som krever tall. Hvis en bruker skulle gjøre en feil bør grensesnittet gi enkel, beskrivende og spesifikk tilbakemelding for å håndtere feilen.

6. **Tillate enkel reversering av handlinger (Permit easy reversal of actions).**

Alle handlinger bør være reverserbare, så langt mulig. Dette letter frykten for å gjøre noe feil og oppfordrer til å utforske ukjente funksjoner.

7. **Hold brukerne i kontroll (Keep users in control).**

   Erfarne brukere ønsker sterkt å føle at de har kontroll på grensesnittet og at grensesnittet reagerer på handlingen deres. De liker ikke overraskelse eller endringer av kjente handlinger, og irriteres av vanskeligheter med å skaffe seg nødvendig informasjon og manglende evne til å få ønsket resultat.

8. **Redusere korttidshukommelse belastning (Reduce short-term memory load).**

   Menneskers begrensning til å prosessere informasjon i kort-tid hukommelse krever at designere unngår grensesnitt hvor brukeren må huske informasjon fra et skjermbilde og så bruke den informasjonen på et annet skjermbilde.

2.1.2 Gestaltprinsippene

Gestaltprinsippene er et sett med lover som stammer fra 1920-tallets psykologi. De beskriver hvordan mennesker vanligvis ser på objekter ved å gruppere lignende elementer, gjenkjenne mønstre og forenkle komplekse bilder (The Interaction Design Foundation, u.d.).

Eleana Gkogka (Gkogka, u.d.) skriver, for Muzli magazine, at gestaltprinsippene er bygd på fire sentrale ideer:

1. **Fremvekst (Emergence):** Den generelle skisserte formen til et objekt blir først sett. Hjernen gjenkjenner simple, veldefinerte objekter raskere enn detaljerte.
2. **Tingliggjøring (Reification):** Mennesker gjenkjenner objekt selv om det mangler deler.
3. **Multi-stabilitet (Multi-Stability):** Tvetydige objekter blir ofte tolket på flere måter og hjernen vil hoppe frem og tilbake mellom alternativene for å være sikker. Som resultat vil et av alternativene bli mer dominerende og det andre blir vanskeligere å se.
4. **Invarians (Invariance):** Folk kan gjenkjenne enkle objekter uavhengig av rotasjon, skala og oversettelse. Hjernen kan oppdage gjenstander fra ulike perspektiver, til tross for forskjellige utseende.

Videre nevner hun syv Gestaltprinsipper som er med på å forme dagens grensesnitt design; Nærhet, Fellesområde, Likhet, Avslutning, Symmetri, Fortsettelse og Felles Skjebne.
Uten å gå i detaljer, gir vi en kort oppsummering for å gi forståelse for hva som ligger bak prinsippene:

1. **Nærhet (Proximity):** Objekt som er plassert nært hverandre blir oppfattet som mer relatert enn de som er plassert lenger vekk.
2. **Fellesområde (Common Region):** Lignende nærhet; objekt plassert i samme region blir oppfattet som gruppert.
3. **Likhet (Similarity):** Objekt som deler lignende visuelle karakteristikk oppfattet som mer relatert enn de som ikke deler karakteristikk.
4. **Avslutning (Closure):** En gruppe objekter er ofte oppfattet som en singel gjenkjennelig form eller figur.
5. **Symmetri (Symmetry):** Symmetriske objekt blir ofte oppfattet som at de tilhører sammen, uansett distanse mellom.
6. **Fortsettelse (Continuation):** Objekter arrangert i en linje eller en myk kurve er oppfattet som mer relatert enn de som er arrangert tilfeldig eller i en hard linje.

7. **Felles Skjebne (Common Fate):** Objekt som beveger seg i samme retning er oppfattet som mer relatert enn de som beveger seg i forskjellige retning.

2.2 Metoder

Når man arbeider med programvareutvikling er det primært to metoder å arbeide med prosjekter på: Plandrevet utvikling og agil utvikling.

2.2.1 Plandrevet utvikling

Plandrevet utvikling (Wikiversity contributors, 2019) er kjent som «tungvekt» eller «tradisjonell» metodikk.

Fullstendige krav kan bli gitt ved oppstart, eller prosjektet kan begynne med generelle ideer som blir mer og mer definert etter hvert intervall. Kravene er prioritert, jo høyere prioritet et krav har jo raskere skal det iverksettes i et intervall. Det vil si at ny funksjonalitet som blir lagt til ikke burde være viktigere enn de som allerede er i prosjektet.

De mest fremtredende plandrevne metodene følger en trinnvis modell; systemet blir analysert, designet, utviklet og testet i intervaller som er klart definert. Hovedfunksjonene blir startet med under første intervall, og ved siste intervall er det fullstendige prosjektet utgitt. Det legges vekt på en byggestein tilnærming, som betyr at det operasjonelle systemet er produsert raskere. Fordelene med denne typen produksjon er:

- Rask ferdigstilling av en fungerende modell
- Gradvis introduksjon av nytt produkt til klienten
- Tidlig og hyppig tilbakemelding fra klienten
- Tidlig funn av stor risiko
- Tillater endringer

En annen modell er fossefallsmodellen; den benytter seg av en lineær struktur og definerer rekkefølgen deloppgaver må utføres. Og følger derfor ikke prinsippene til iterativ utvikling. Modellen planlegges fra starten til sluttresultatet av prosjektet og kan bli brutt ned i 5 steg:

- Kravanalyse
- Design
- Koding og iverksettelse
- Testing
- Vedlikehold

Tidligere faser må bli fullført så nøyaktig som mulig, denne modellen er strukturert slik at man ikke faller tilbake til et tidligere steg. Feil i tidlige faser av utviklingen vil resultere i økt kostnad, tid og innsats krevd av teamet.

Den største ulempen er den statiske strukturen hvor denne metoden ikke kan bli benyttet i et dynamisk miljø som krever iterativ fremgang.

2.2.2 Smidig utvikling (Agile)

Smidig utvikling er et samlebegrep for alle utviklermetoder som baserer seg på verdiene og prinsippene i det smidige manifest:

«Dette vil si: Selv om punktene som står til høyre har verdi, så verdsetter vi punktene til venstre enda høyere» (Beck, et al., u.d.).

Den smidige utviklingsmetodikken er basert på å lage mange funksjonelle versjoner av systemet og inkludere oppdragsgiveren i utviklingsprosessen. Denne utviklingsprosessen er mest effektivt brukt i prosjekter med noe usikkerhet, eller der omfanget av oppgaven er noe usikkert. En agil utviklingsmetodikk er veldig fleksibel for endringer og vil oftest resultere i et sluttprodukt oppdragsgiveren er svært fornøyd med grunnet stor påvirkningskraft på produktet under utviklingsprosessen.

Oppgavene i utviklingsprosessen vil bli tildelt en viktighetsgrad og en poengsum etter vanskelighetsgrad. Vanskelighetsgraden blir bestemt innad i utviklergruppen og brukes for å beregne effektiviteten av utviklingen i et møte i slutten av utviklingstiden for iterasjonen. Dette utviklingsvinduet er vanligvis en uke eller to, men kan være kortere eller lengere basert på kompleksitet og usikkerhet. Det er vanlig å ha uformelle møter nesten hver dag for å få oversikt over hvordan utviklingen ligger an i forhold til fremdriftsplanen (Pfahl, 2014).

Sommerville (Sommerville, 2016, p. 74) sier:

> «Agile metoder er intervallbaserte utviklingsmetoder hvor intervallene er korte, og vanligvis blir det gitt ut nye utgivelser hver andre eller tredje uke som er gjort tilgjengelig for kunden. De involverer kunden i utviklings prosessen for å få rask tilbakemelding på kravendringer. Metodene minimerer dokumentasjon ved å bruke uformell kommunikasjon i stedet for formelle møter med skriftlig dokumentasjon.
> Dens tilnærming til programvareutvikling anser design og implementasjon som sentrale aktiviteter i programvareprosessen.»

2.3 Datasikkerhet

For en web-applikasjon er det viktig å sikre mot uautorisert tilgang og minimere sjansen for angrep mot tjenestene. Det er viktig å sikre sluttbrukeren, og unngå uønsket tilgang til sensitiv informasjon. Derfor bør systemet iverksette en eller flere av metodene beskrevet under.

2.3.1 Hashing

Fra boken «Principles of Information Security» (Whitman & Mattord, 2017, p. 465) av Whitman og Mattord blir hash funksjoner beskrevet som matematiske algoritmer som blir brukt til å bekrefte identiteten til en spesifikk melding og bekrefte at innholdet ikke er blitt endret. Videre forklarer de at hash algoritmer brukes for å lage en hash verdi, også kjent som «message digest», ved å konvertere meldinger av ulike lengder til et sammendrag med fast lengde.

Hashing er ansett som en en-veis operasjon på den måten at meldingen alltid vil gi samme hash verdi, men hash verdien selv ikke kan bli brukt til å fastslå innholdet av meldingen. Dette gjør at hash verdier er brukt til passordverifiseringssystem for å bekrefte identiteten til en bruker. Verdien blir kalkulert ut ifra det originalt utstedte passordet og verdien er den som blir lagret for sammenligning senere. Når brukeren logger på senere vil systemet kalkulere en hash verdi basert på det passordet brukeren skriver inn og sammenligner den med den lagrede verdien for å bekrefte identitet.

2.3.2 Salt

«Salting» (Gordon & Hernandez, 2016, p. 360) er brukt til å sikre lagrede passord. Det er tilfeldig data som er lagt til hash-verdien for å beskytte det mot å bli lest av systemet.

En ny salt er tilfeldig generert for hvert enkelt passord. I de fleste tilfeller er salten og passordet sammenkoblet og bearbeidet med en hash-funksjon og resultatet blir lagret i en database.

Salting beskytter mot ordbokangrep eller dets hashede motpart «rainbow table attack». Et ordbordangrep sjekker de hashede verdiene mot kjente hashede passord som en ordbok. De hashede ordene er enten kalkulert i forkant eller hentet fra eksisterende passordangrep.

2.3.3 Sosial manipulering

Fra Wikipedia (Wikipedia contributors, 2019): «Sosial manipulering er en metode for å bruke psykologisk manipulering for å overbevise folk om å gi fra seg konfidensiell informasjon, som brukernavn og passord.»

En av de mest kjente formene for sosial manipulering er phishing (Whitman & Mattord, 2017, p. 82); hvor en angriper gir ut noe som kan se ut som en legitim melding, ofte e-post, som inneholder skjult eller innebygd kode som omdirigerer svar til en tredjepartsside for å få konfidensiell informasjon.

2.3.4 SQL injeksjon

SQL-injeksjon (w3schools, u.d.) er en teknikk som kan ødelegge en database. Det er en av de mest vanlige web-hacketeknikkene som finnes. Teknikken plasserer skadelig kode i en SQL-spørring gjennom nettsidens innfyllingsfelt.

SQL-injeksjon skjer vanligvis når nettsiden spør etter brukerinnfyllingsfelt, som når man skal logge inn med brukernavn og passord, men i stedet blir en SQL-spørring sendt inn. Med det kan data i databasen bli endret og data som ikke skal bli sett, blir tilgjengelig.

2.3.5 Andre trusler

Ikke alle trusler er ment som angrep mot systemet eller
applikasjonen. Menneskelig svikt, maskinvaresvikt og naturkrefter er trusler som også er til stede.
Hvis vi lager systemet med hensyn til prinsippene nevnt over, så lager vi systemet rundt at brukeren ikke skal kunne gjøre alvorlige feil.

2.4 Database

I alle virksomheter er informasjonsflyten utfordring, og i mange bedrifter flyter informasjon enten fysisk eller digitalt. Det som oppleves i mange virksomheter er dårlig informasjonsflyt og derfor er databaser essensielt. Databaser standardiserer hvordan informasjon lagres, og tillater søkefunksjoner og bedre strukturering av dataen. Behovet varierer fra bedrift til bedrift, men er størst i bedrifter hvor det er flere typer data som skal lagres og kategoriseres.

Bratbergsengen (Bratbergsengen, u.d.) skriver for Store Norske Leksikon at en database er en samling data lagret på et elektronisk medium. Datasamlingen er organisert og strukturert etter en bestemt strategi eller modell - en databasemodell.

Databasen inneholder unike ressursenheter, poster eller dataobjekter, som mange brukere konkurrerer om å få tilgang til (lese, skive eller oppdatere). Dette skiller et dataobjekt fra andre typer ressurser som lagringsplass og CPU-kapasitet. Et dataobjekt kan ikke erstattes, og databasesystemet må derfor sørge for at konkurransen om dataobjektene, systemfeil eller ytre hendelser ikke fører til at data går tapt eller får feil verdi.

Den mest populære databasemodellen er relasjonsmodellen. Alle operasjoner på relasjonsmodellen er definert i en standard som kort kalles SQL-standarden.

Eksempler på systemer som følger relasjonsmodellen er: Ingres, Oracle, SyBase, DB2, SQL Server og MySQL.

2.5 Webteknologi

Nettleserne benytter seg av forskjellige teknologier for å vise nettsider, dette innebærer HTML, CSS JavaScript og AJAX, men det finnes mange flere. Funksjonen av de forskjellige teknologiene er kort forklart å sette opp strukturen av nettsiden, endre utseende og legge til funksjonalitet.

2.5.1 HTML

HyperText Markup Language (HTML) (W3C, 2016) er et standard markeringsspråket som brukes for å beskrive strukturen av en nettside.

Fra Wikipedia (Wikipedia contributors, 2019):

> «Nettlesere mottar HTML dokumenter fra en webserver, eller fra lokal lagring, og gjengir dokumentet i multimedia nettsider.
>
> HTML elementer er byggeklossene til HTML sider. Med HTML kan bilder og andre objekter som interaktive skjemaer bli innebygd i den gjengitte siden. HTML gir en mulighet til å lage strukturerte dokument ved å betegne strukturelle semantikk for tekst som overskrifter, paragrafer, lister osv».

2.5.2 CSS

Fra W3C (W3C, 2016): «Cascading Style Sheet (CSS) er språket for å beskrive presentasjonen av nettsider, med bruk av farge, oppsett og skrifttyper. Det tillater oss å utvikle presentasjonen til flere typer enheter, som store skjermer, små skjermer eller printere. CSS er uavhengig av HTML og kan brukes med hvilket som helst XML-basert markeringsspråk».

2.5.3 JS

JavaScript (JS) (Wikipedia contributors, 2019) er et høynivå tolket skriptspråk. JS muliggjør interaktive nettsider og er en essensiell del av nettapplikasjoner. De aller fleste nettsider bruker JavaScript, og de fleste nettlesere har en dedikert JavaScript-motor for å kjøre det.

2.5.4 AJAX

Asynchronous JavaScript and XML (AJAX) (Wikipedia-brukere, 2016) er en webutviklingsteknikk for å lage interaktive nettsider. Det gjøres ved at nettsidene utveksler litt og litt data med serveren i bakgrunnen i stedet for å laste hele siden på nytt hver gang brukeren gjør en endring.

2.5.5 Python

Python er et objektorientert programmeringsspråk med lettlest og klar syntaks.

I Python deles koden opp etter innrykk og ikke etter spesialtegn som {} (Wikipedia-brukere, 2019).

Python blir brukt til webutvikling, programvareutvikling, matematikk og systemskripting. Det kan brukes på servere for å lage webapplikasjoner, ved siden av programvare for å lage arbeidsflyt, til å håndtere store mengder data og utføre kompleks matematikk, til å koble til databasesystemer. Det kan brukes for rask prototyping eller for produksjonsklar programvareutvikling.

Python fungerer på forskjellig plattformer (Windows, Mac, Linux, Raspberry Pi osv.). Det kjøres på et tolksystem, som gjør at koden kan kjøres så snart den er skrevet (w3shools, u.d.).

2.5.6 Flask

Flask er et mikrorammeverk for Python basert på Werkzeug, Jinja 2 og gode intensjoner, sier de selv på sin hjemmeside (Armin Ronacher, u.d.).

Flask sitt mål er å holde kjernen simpel, men utvidbar. Dette gir brukerne mer frihet til å velge selv hvilke funksjoner som trengs for applikasjonen (Anon., u.d.).

Flask brukes for å sette opp nettbaserte applikasjoner på en effektiv og rask måte. Det baserer seg på at kun nødvendige pakker er importert og inkludert i biblioteket og gjør servergrensesnittet mer fleksibel. Utvikling for nye teknologier kan bruke nye eller mer spesialiserte pakker.

2.5.7 MySQL

MySQL leverer en veldig rask, flertråds, flerbruker og robust SQL database server. Serveren er ment for kritisk, tunglastede produksjonssystemer, samt for å legge inn i masseutviklet programvare (MySQL™, 2019).

MySQL er det mest populære åpen kildekode SQL databasehåndteringssystemet (MySQL™, 2019).

3 MATERIALER OG METODE

3.1 Metode

3.1.1 Utviklingsmetode

For dette prosjektet er det blitt brukt en tilnærmet plandreven utviklingsmetode (Kap. 2.2.1) hvor vi delte trinnene inn i to-ukers intervaller. Hvert intervall ble startet med et gruppemøte. I gruppemøtet ble det diskutert hva som skulle gjøres og ting som måtte bli tatt opp med oppdragsgiver.

Første møtet ble med veileder, Di Wu, for å diskutere oppgaven gitt av oppdragsgiver, Bjørn Erik Melland. Der planla vi spørsmål som skulle gi oss et overblikk over hva som var ønsket ut av prosjektet.

I vårt første møte med oppdragsgiver fikk vi en videre forklaring av oppgaven. Oppdragsgiver fikk forklart hva han ville ha og vi fikk stilt noen generelle spørsmål for å komme i gang med prosjektet. Vi startet med å skrive ned kravspesifikasjon og rangerte hvilke krav som var kjernefunksjoner og hvilke funksjoner som skulle utvikles videre ut fra dem. Det ble enighet om at det å lage en nettside ville være den beste løsningen for prosjektet.

Etter vårt første møte med oppdragsgiver hadde vi et nytt gruppemøte for å tegne opp forslag til hvordan nettsiden skulle se ut, dette ble gjort både på tavle, papir og nettsiden draw.io. Oppdragsgiver hadde nevnt at det skulle 'rope' AaFK av den, så vi valgte å ta inspirasjon av nettsiden til AaFK; aafk.no**.** Se *Vedlegg 3 Møtereferater* for mer informasjon.

I starten av hvert intervall ble det diskutert hva som hadde blitt gjort og hva som ikke hadde blitt gjort. Forslag til endringer og planer for hva som skulle være i fokus neste intervall ble lagt frem.

I begynnelsen ble det avholdt hyppige møter med oppdragsgiver for å fastlegge designet på nettsiden. Etter hvert som designet var på plass, ble møtene satt opp kun for å vise fremskritt og for at oppdragsgiver skulle kunne ta avgjørelser og komme med nye eventuelle ideer etter forrige møte.

I starten av prosjektet fulgte vi smidig utviklingsmetoder med hyppige møter og intervaller for å ferdigstille deler av et system. Dette viste seg å ikke være den mest effektive arbeidsformen for vår gruppe, og ettersom vi naturlig hadde god dialog følte vi

det var fordelaktig å bytte til en plandrevet utviklingsmetode. Ved å endre utviklingsmetodikk klarte vi å lage lengre planer og sette bedre, mer definerte delmål.

3.1.2 Programmeringsspråk

Siden det ble bestemt at prosjektet skulle være en nettside er HTML og CSS (Kap. 2.5.1 og 2.5.2) nesten unngåelig. MySQL (Kap. 2.5.7) blir brukt for å lage en database som kan holde på all dataen. Python (Kap. 2.5.5) er brukt for å koble databasen og HTML-dokumentene. Flask er brukt for å gjøre HTML dokumentene mer ryddig, ved å lage et HTML-dokument som holder på hovedfunksjonene og henter inn fra andre HTML-dokument, og sikrere med noen fine utvidelser (Json, Werkzeug, SQLAlchemy, etc.). Flask er også modulært og tillater derfor stor frihet rundt valg av løsninger. JavaScript (Kap. 2.5.3) er brukt for å gjøre mer avanserte funksjoner som oppdatering av innhold uten å laste inn siden på nytt.

Valg av programmeringsspråk kommer av at Python og MySQL er kjente og mye brukt. Begge språkene er åpen kildekode, støtter mange forskjellige plattformer, og har en stor og aktiv brukerbase.

3.2 Bibliotek

3.2.1 JQuery

JQuery (w3schools, u.d.) er et bibliotek i JavaScript som forenkler bruken av JavaScript på nettsiden, ved å ha metoder som gjør at vi slipper å skrive mange linjer med kode for vanlige oppgaver. Det har et godt beskrivende motto som sier: «Write less, do more».

3.2.2 DataTable

DataTable er en plug-in for jQuery-biblioteket. Det er et svært fleksibelt verktøy, som holder på mange nyttige egenskaper som kan gjøre HTML-tabeller mer brukervennlige (SpryMedia Ltd., u.d.).

3.2.3 WTForms

WTForms (Johansson & Crasta, u.d.) er et fleksibelt skjema validering- og visningsbibliotek for Python web utvikling.

3.2.4 Werkzeug

Werkzeug (Ronacher, u.d.) er omfattende WSGI (Web Server Gateway Interface) web-applikasjonsbibliotek. Flask er basert på Werkzeug og bruker det til å håndtere WSGI, samtidig som det gir mer struktur og mønstre for å definere kraftige applikasjoner.

3.2.5 SQLAlchemy

SQLAlchemy (Bayer, u.d.) er en verktøykasse og ORM (Object RelationalMapper) som gir utviklere den fulle effekten og fleksibiliteten av SQL. Den gir en komplett pakke med kjente mønstre designet for effektiv og høyt ytende databasetilgang, tilpasset et simpelt og Pytonsk domenespråk.

3.3 Verktøy

3.3.1 PyCharm

PyCharm er et utviklingsmiljø spesielt designet for utvikling av Python prosjekt.

3.3.2 Git

Git er et versjonskontrollsystem som gir oss muligheten til å være flere som arbeider med samme prosjekt samtidig og hver for oss. Det tillater oss å ha en egen versjon av

prosjektet på vår egen datamaskin som kan bli sammenflettet med master-grenen til prosjektet. Når en endring blir gjort til master-grenen kan dette bli sendt til en felles database de andre utviklerne kan hente gren-endringene fra denne versjonen.

Hver gang det blir gjort en endring i en av grenene i systemet blir endringene i prosjektet lagret, slik at vi kan gå tilbake hvis problem skulle oppstå med nyere versjoner av prosjektet.

3.3.3 Bitbucket

Bitbucket (contributors, 2019) er en nettbasert versjonskontrollsoppbevaringssted for kildekode og utviklingsprosjekt som bruker enten Mercurial eller Git. Bitbucket tilbyr både kommersielt og gratis bruk. Kommersielt bruk inkluderer funksjoner som er gunstig for større prosjekter, mens gratis bruk er mer rettet mot mindre grupper og prosjekter. Et annet alternativ kan være for eksempel Github, som hovedsakelig bruker Git.

3.3.4 SourceTree

SourceTree er et grafisk brukergrensesnitt for å navigere og bruke Git, enten lokalt eller gjennom eksterne oppbevaringssteder.

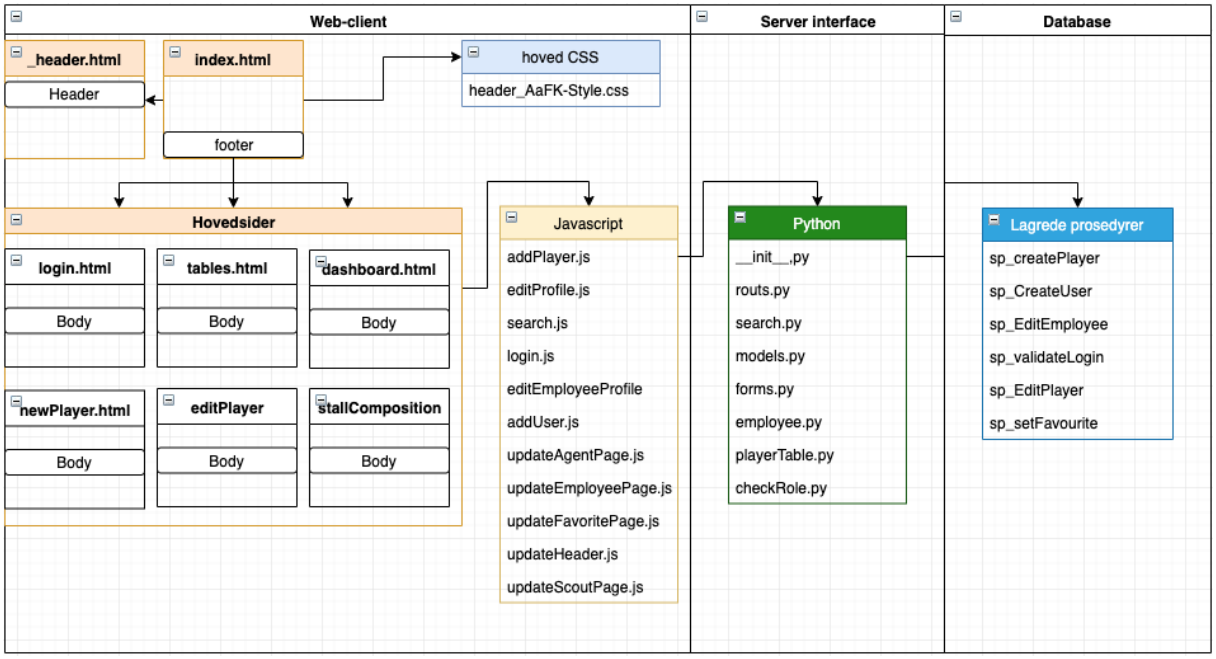
3.3.5 MySQL Workbench

MySQL (MySQL™, u.d.) beskriver Workbench veldig enkelt selv: «MySQL Workbench er et forent visualiseringsverktøy for databasearkitekter og utviklere. Det tilbyr datamodellering, SQL utvikling og omfattende administreringsverktøy for serverkonfigurasjon, administrasjon av brukere, sikkerhetskopiering og mer».

4 RESULTATER

4.1 Konseptmodell

Under er en konseptuell modell over webapplikasjonen og interaksjonen med server-grensesnittet og databasen. Dette er en overordnet og noe forenklet struktur, men den viser hovedtrekkene i applikasjonen og kommunikasjonen mellom de forskjellige leddene i applikasjonen.



Figur 1: Konseptmodell Applikasjon

4.1.1 HTML struktur

Jinja2 tillater implementert segmentering av HTML kode. Dette betyr at alle HTML-filene kan arve oppsett og felles funksjoner fra samme overordnede HTML-fil.

Sidene inneholder JavaScript filer for å gjøre kall til servergrensesnittet, slik at servergrensesnittet kan returnere verdier fra databasen eller benytte de lagrede prosedyrene.

Figur 2: HTML Document Structure (CodesCracker, u.d.)

Prosjektet bruker Flask som er basert på Jinja2 og tilbyr mange verktøy for å sette opp nettsider.

Alle nye sider henter informasjon fra samme sted. Dette stedet er index.html, her lager vi en seksjon som holder på all informasjon som skal inkluderes i de forskjellige sidene.

Jinja2 lar oss lage en base hvor vi kan samle alt HTML-filene skal inneholde; en generell CSS fil, script og navigasjonsbar. Alle HTML-filer er koblet til denne base HTML-filen ved bruk av kun noen korte linjer, som vist under i figur 3.

Jinja2 har mange fine små funksjoner som for eksempel å endre fanetittelen med en «if-setning» som sjekker om vi har satt en tittel for siden i vårt kall i routes.py filen, hvis ikke setter den standardtittelen «AaFK player management».

Måten det er gjort på er at hver ny side inkluderer {% extends index.html %} og {% block content %}{% endblock %}, hvor "content" er en beskrivende tittel for hva den skal vise.

```
<!DOCTYPE html>
<html lang="en">
```

```
<head>
   <meta charset="UTF-8">
   {% if title %}
      <title>AaFK {{ title }}</title>
   {% else %}
      <title> AaFK player management</title>
   {% endif %}
   <link rel="stylesheet" type="text/css" href="../static/Header_AaFK-Style.css">
   <link rel="stylesheet" type="text/css" href="../static/Employee-AaFK-Style.css">
   <link rel="stylesheet" type="text/css" href="../static/User-Dashboard_AaFK-Style.css">
   <link rel="stylesheet" type="text/css" href="../static/Player-Profile_AaFK-Style.css">
   <link rel="stylesheet" type="text/css" href="../static/newPlayerForm_AaFK-Style.css">
   <link rel="stylesheet" type="text/css" href="../static/Login_AaFK-Style.css">
   <link rel="stylesheet" type="text/css" href="../static/Add-User_AaFK-Style.css">
   <link rel="stylesheet" type="text/css" href="../static/Tables.css">
   <script src="http://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"
type="text/javascript"></script>
   <script src="https://cdn.datatables.net/1.10.19/js/jquery.dataTables.min.js"
type="text/javascript"></script>
   <script src="../static/js/logIn.js"></script>

</head>


<body>

{% include 'includes/_header.html' %}
{% block userHome %}{% endblock %}
{% block error %}{% endblock %}
{% block dashboard %}{% endblock %}
{% block profile %}{% endblock %}
{% block tables %}{% endblock %}
{% block newPlayer %}{% endblock %}
{% block login %}{% endblock %}
{% block addUser %}{% endblock %}
{% block scoutPage %}{% endblock %}
{% block agentPage %}{% endblock %}
{% block stallComposition %}{% endblock %}
{% block employeeProfile %}{% endblock %}
{% block shadowTeam %}{% endblock %}


</body>

</html>
```

Figur 3: index.html; Base HTML

Login.html trenger ikke inkludere noen «html» eller «body» tag siden den arver det fra "index.html".

```
{% extends 'index.html' %}


{% block login %}

   <main>

      <section class="loginForm">

         <h2>Login</h2>
         <form action="/validateLogin" method="post" novalidate…>
      </section>
   </main>

{% endblock %}
```

Figur 4: login.html

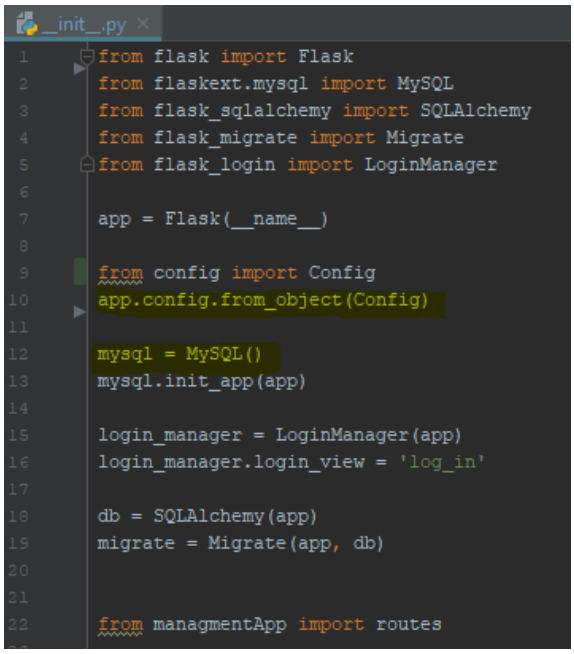
Den arver også CSS filen, som gjør at dens egen CSS fil kun trenger inneholde «klasse» eller «id» tagger.

```
#btnLogIn{…}

#btnLogIn:hover{…}

.loginForm{…}

.container {…}

span.psw {…}

/* Change styles for span button on extra small screens */
@media screen and (max-width: 300px) {…}
```

Figur 5: Login_AaFK-Style.css

4.1.2 Grensesnitt til database

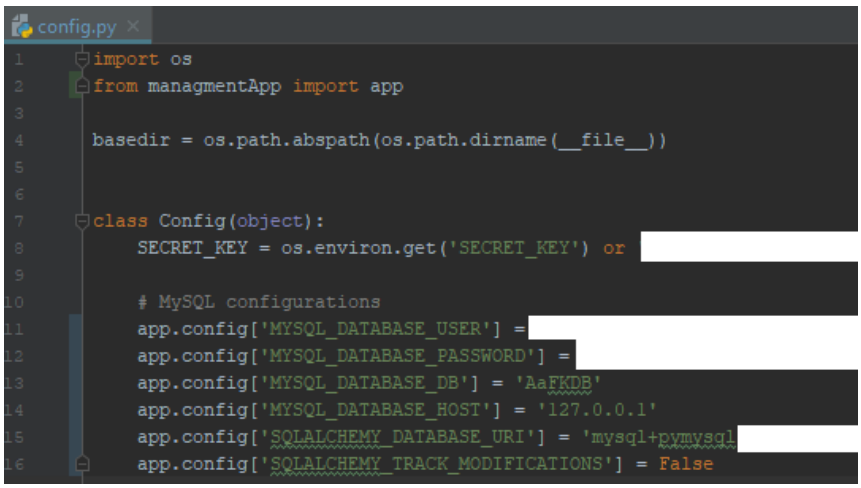
Gjennom «__init__.py» setter vi opp systemet slik at det er enkelt å koble seg til databasen for funksjoner som trenger det.



```python
from flask import Flask
from flaskext.mysql import MySQL
from flask_sqlalchemy import SQLAlchemy
from flask_migrate import Migrate
from flask_login import LoginManager

app = Flask(__name__)

from config import Config
app.config.from_object(Config)

mysql = MySQL()
mysql.init_app(app)

login_manager = LoginManager(app)
login_manager.login_view = 'log_in'

db = SQLAlchemy(app)
migrate = Migrate(app, db)


from managmentApp import routes
```

Figur 6: __init__.py, initialisering filen

I vår «__init__.py» fil, hvor vi initierer applikasjonen, kaller vi på en klasse «Config». Klassen holder på konfigurasjonsinformasjon, hvor tilkoblingsinformasjonen til databasen ligger.

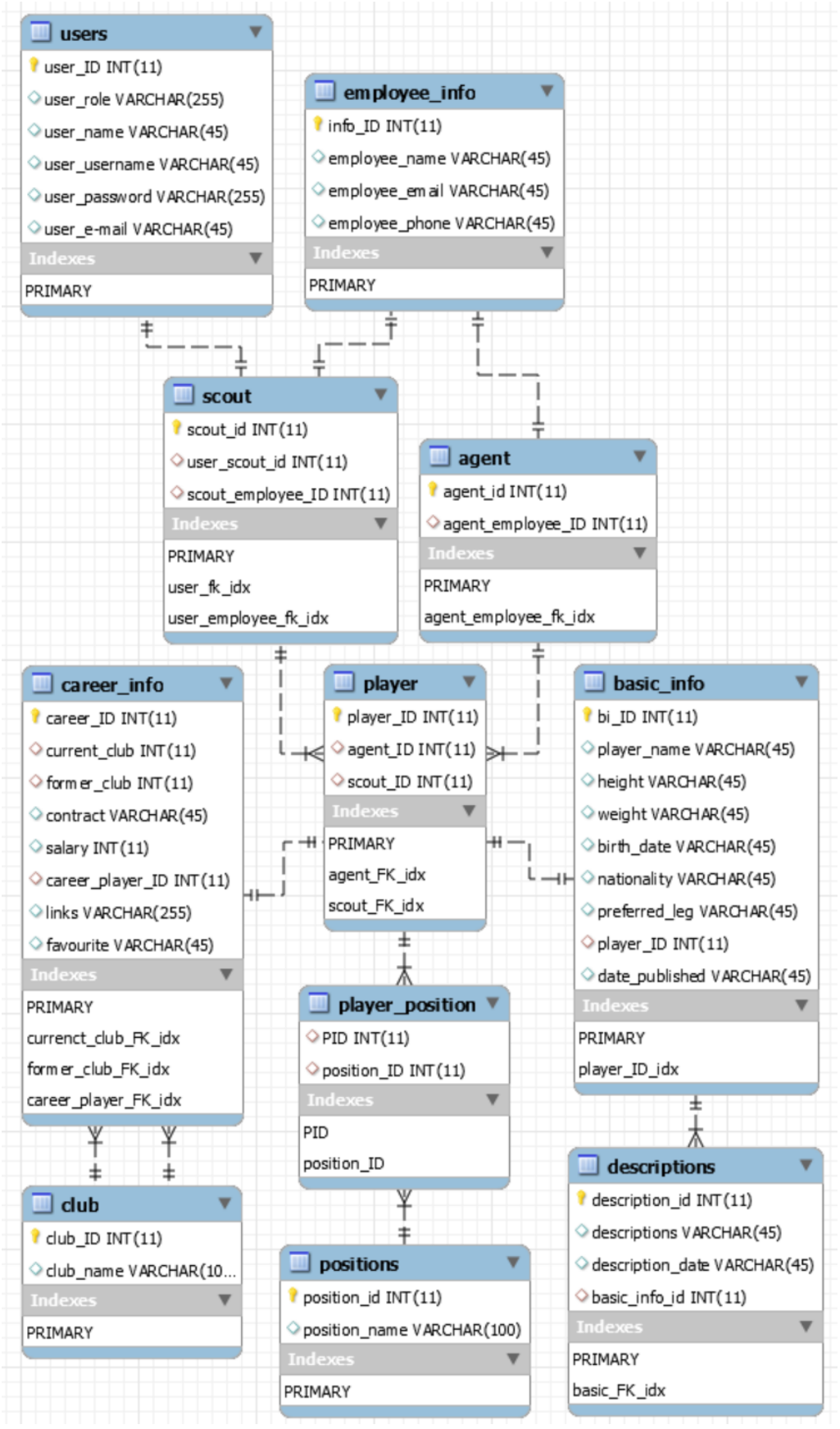


```python
import os
from managmentApp import app

basedir = os.path.abspath(os.path.dirname(__file__))


class Config(object):
    SECRET_KEY = os.environ.get('SECRET_KEY') or

    # MySQL configurations
    app.config['MYSQL_DATABASE_USER'] =
    app.config['MYSQL_DATABASE_PASSWORD'] =
    app.config['MYSQL_DATABASE_DB'] = 'AaFKDB'
    app.config['MYSQL_DATABASE_HOST'] = '127.0.0.1'
    app.config['SQLALCHEMY_DATABASE_URI'] = 'mysql+pymysql
    app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
```

Figur 7: config.py, Konfigurasjonsfilen

«__init__.py» er et standard klassenavn for en startklasse, så om ikke annet er spesifisert vil kompileringen lete etter en «__init__.py» fil.

4.2 Databasemodell



Figur 8: Databasestrukturmodell fra MySQL Workbench

Prosjektet er gjennomført med sikte på å tillate videre utvikling. Alle tabell- og attributtnavn er derfor valgt utfra en tanke om at databasen skal framstå så logisk og intuitiv som mulig. Dette er også grunnen til at alle navn i databasen er skrevet på engelsk. Nedenfor er alle navn oversatt til norsk.

4.2.1 Attributter

Spillertabellen
Har tre attributter: en unik spiller_ID, en agent_ID og en speider_ID.

- **spiller_ID** - Har en automatisk økning, som sørger for at brukeren ikke trenger å fylle inn slik triviell informasjon, forhindrer brukerfeil og gjør det enkelt å sortere etter nyeste eller eldste entitet.
- **agent_ID** - Sørger for at en spiller kan være knyttet til en agent, mens en agent kan være knyttet til flere spillere.
- **speider_ID** - Knytter en spiller til den speideren som opprettet spilleren.

Attributten "spiller_ID" er en primærnøkkel, alle tabellene utenom tabellen "spiller-posisjon" holder på en tilsvarende attributt.

Agenttabellen
Har én attributt: "agent_ansatt_ID". Attributten er en fremmednøkkel for tabellen "ansatt_informasjon" og sørger for at applikasjonen kan hente ut kontaktinformasjon om en agent.

Speidertabellen
En speider er knyttet til entiteten "ansatt_informasjon" og én bruker. Derfor holder den på fremmednøklene "bruker_speider" og "speider_ansatt".

Ansattinformasjonstabellen
En entitet i tabellen "ansatt_informasjon" er i stand til å holde på kontaktinformasjon for en speider eller ansatt. Den holder på attributtene: "ansatt_navn", "ansatt_epost" og "ansatt_telefon".

- **ansatt_navn** - Holder på navnet til en speider eller agent.
- **ansatt_epost** - Holder på en e-postadresse.
- **ansatt_telefon** - Holder på et telefonnummer. Attributten lagres som en VARCHAR, slik at den også kan holde på retningsnummer.

Brukertabellen
En bruker har fem entiteter: "rolle", "navn", "brukernavn", "passord" og "e-post".

- **rolle** – Bestemmer hvilke rettigheter en bruker har. Den kan holde på administrator eller speider, men er tilrettelagt for mulighet til å legge til flere roller om ønskelig.
- **navn** - Holder på det fulle navnet til en bruker.
- **brukernavn** - Holder på brukernavnet.
- **passord** - Holder på passordet til brukeren.
- **e-post** - Holder på brukerens e-postadresse.

Karriereinformasjonstabellen
Inneholder attributter som beskriver en spillers klubbhistorikk og kontraktinformasjon. Den har syv attributter: "nåværende_klubb", "tidligere_klubb", "kontrakt", "lønn", "karriere_spiller_ID", "lenker" og "favoritt".

- **nåværende_klubb** - Koblet til klubbtabellen og gir IDen til den klubben spilleren spiller for.

- **tidligere_klubb** - Er også knyttet til klubbtabellen og holder på IDen til den forrige klubben spilleren spilte for.
- **kontrakt** - Viser utløpsdatoen til spillerens kontrakt.
- **lønn** - Holder på spillerens årlige inntekt. Innholdet av denne attributten kan anses som sensitiv informasjon, og bør behandles deretter.
- **karriere_spiller_ID** - Knytter en spiller til karriereinformasjonstabellen.
- **lenker** - Holder på lenker til nettsider som inneholder detaljert info om den aktuelle spilleren.
- **favoritt** - Brukes til å kategorisere en spiller som favoritt eller ikke favoritt.

Klubtabellen
Har én attributt: "klubb_navn". Attributten holder på navnene til alle de forskjellige klubbene registrert i databasen.

Grunnleggendeinformasjonstabellen
Holder på informasjon om en spillers fysikk og annen grunnleggende informasjon om en spillerentitet. Denne informasjonen lagres i åtte forskjellige attributter: "navn", "høyde", "vekt", "fødselsdato", "nasjonalitet", "foretrukket_fot", "spiller_ID" og "dato_publisert".

- **navn** - Holder på en spillers fulle navn.
- **høyde** - Holder på en spillers høyde i centimeter.
- **vekt** - Holder på vekten til en spiller i kilogram.
- **fødselsdato** - Holder på spillerens fødselsdato.
- **nasjonalitet** - Holder på landet spilleren er fra eller har valgt å representere internasjonalt.
- **foretrukket_fot** - Holder på den foten som er dominerende for spilleren.
- **spiller_ID** - Kobler en grunnleggendeinformasjonentitet til en spillerentitet.
- **Dato publisert** - Holder på den datoen spiller-entiteten først ble registrert i databasen.

Beskrivelsestabellen
Denne tabellen holder på to attributter: "beskrivelser" og "beskrivelse_dato". Attributten "beskrivelser" inneholder en kort beskrivelse av spilleren, fylt inn av en speider. "beskrivelse_dato" holder datoen beskrivelsen ble registrert.

Posisjonstabellen
Har én attributt: "posisjonsnavn". Attributten holder på alle registrerte spillerposisjoner, for eksempel spiss eller målvakt.

Spillerposisjonstabellen
Har to attributter: "SID" og "posisjon_ID". SID står for spiller ID og attributten kobler entiteter fra denne tabellen til spillerentiteter. Attributten "posisjon_ID" kobler tilsvarende entiteter fra denne tabellen til posisjonsentiteter.

4.2.2 Forhold

Spillertabellen er i senter av databasen og kobles direkte eller indirekte til alle tabeller. Likevel holder den bare på noen få attributter. Vi har fordelt attributtene over flere forskjellige tabeller for å forbedre forholdene mellom dem, og fordi det å legge til flere, endre eller slette attributter skal føre til færrest mulig komplikasjoner.

En spillerentitet er opprettet av en bruker, enten i form av en speider eller en administrator. Hvis en spiller blir opprettet av en speider skal denne spilleren være koblet til den speideren. En speider kan registrere flere spillere. Derfor har spillertabellen et en-til-mange forhold til speidertabellen og speidertabellen har et en-til-en forhold til brukertabellen.

En spiller kan knyttes til en speider, men også til en agent. En agent kan være representant for flere spillere og det er derfor et en-til-mange forhold mellom agent- og spillertabellen. Både agent- og speidertabellen har sin kontaktinformasjon lagret i ansatt-informasjon-tabellen.

En ansatt kan være en speider, en agent eller begge deler. Flere agenter eller flere speidere skal ikke ha samme kontaktinformasjon. Hver agent eller speider skal ha kun en liste med kontaktinformasjon. Derfor er det et en-til-en forhold mellom både agent- og ansattinformasjonstabellen, og mellom speider- og ansatt-informasjonstabellen.

Spillerentiteter er knyttet mot en karriereinformasjonsentitet. En karriereinformasjonsentitet kan kun knyttes til en spiller. Det samme gjelder forholdet mellom spillerentiteter og grunnleggendeinformasjonsentiteter. Derfor er spillertabellen knyttet til begge disse tabellene med en-til-en forhold.

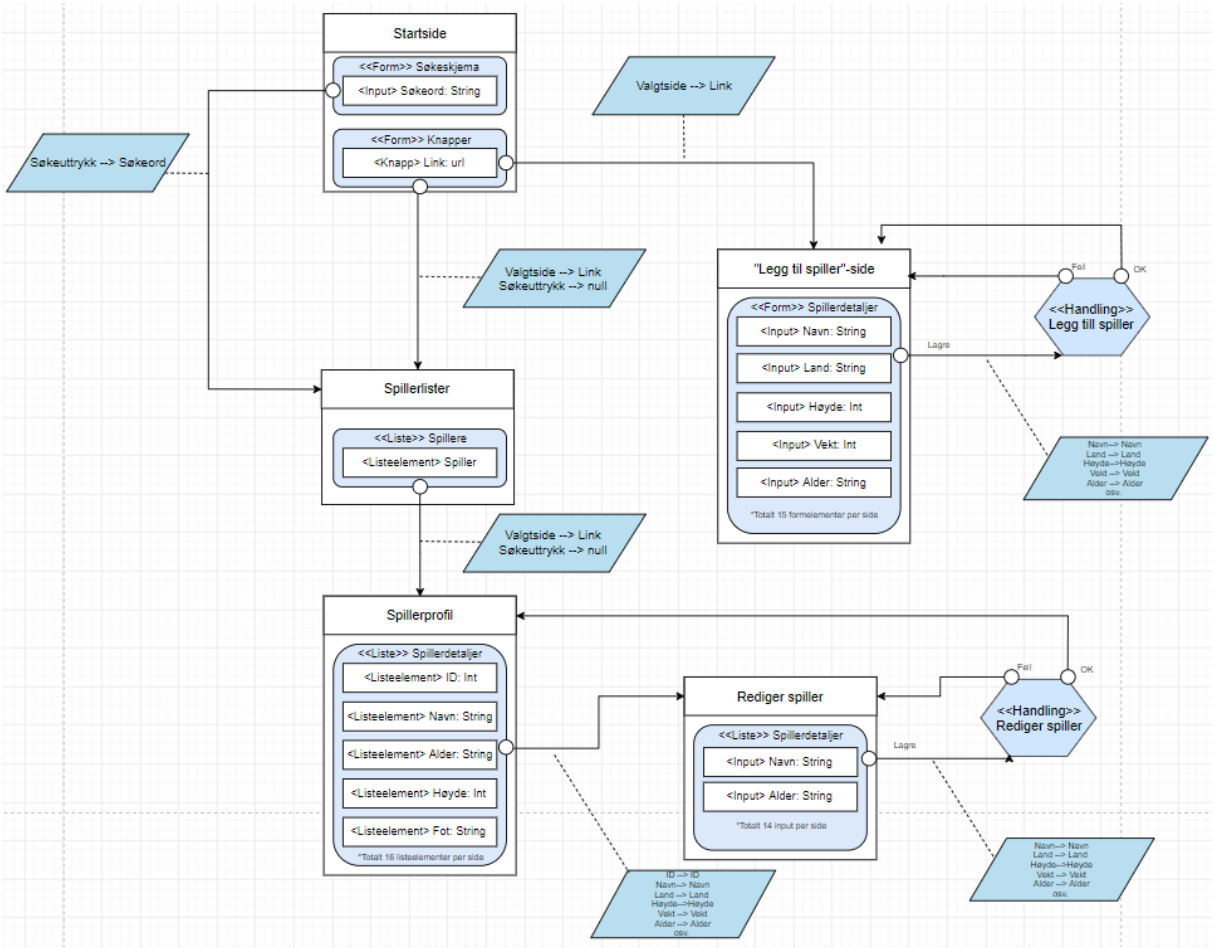
Tabellen karriereinformasjon kobles til klubbtabellen på to steder ved bruk av fremmednøklene “nåværende_klubb” og “tidligere_klubb”. Disse attributtene kobles til kun en klubbentitet hver, mens klubbentiteter kan kobles opp mot flere karriereinformasjonsentiteter. Det er av den grunn en-til-mange forhold mellom karriereinformasjon- og klubbtabellen.

Mellom karriere-informasjon-tabellen og beskrivelsestabellen er det et en-til-mange forhold. Dette er for at det skal være mulig å holde en logg på alle beskrivelsene som er oppført hos en spiller og fordi en beskrivelse skal tilhøre kun en spiller.

En spillerentitet er i stand til å holde på flere posisjoner, og en posisjon kan holdes på av flere spillere. Vi har derfor skapt et kunstig mange-til-mange forhold mellom spillertabellen og posisjonstabellen. Dette har vi gjort ved å lage spiller-posisjon-tabellen som har et mange-til-en forhold til spillertabellen og et en-til-mange forhold til posisjonstabellen.

4.3 Interaksjonsflyt

4.3.1 Interaksjonsflyt ved vanlig bruk



Figur 9: Interaksjonsflytmodell for å se, søke på, redigere og legge til nye spillere.

Modellen ovenfor viser normal interaksjonsflyt for håndtering av spillerdata. Fra startsiden kan en bruker navigere seg til sider for å se, endre eller legge til ny spillerdata.

De forskjellige visningene er representert i modellen ved hvite bokser med tittel. Hver visning holder på dataobjekter. Disse dataobjektene vises i modellen som blå bokser med felt for verdier med datatyper. De små hvite sirklene koblet til disse dataobjektene representerer handlinger. Hver handling peker mot en visning, så hvis denne handlingen blir kalt vil tilsvarende visning bli lastet inn.

De blå parallellogrammene viser hvilke parametere som er tilknyttet en handling.

Modellen beskriver all funksjonalitet for en bruker med rollen Speider. En bruker med rollen Administrator har tilgang til de samme funksjonene, men har også tilgang til flere forskjellige visninger for presentering av spillerdata. Administratorer kan også se og endre informasjon om agenter og speidere.

4.4 Grafisk brukergrensesnitt

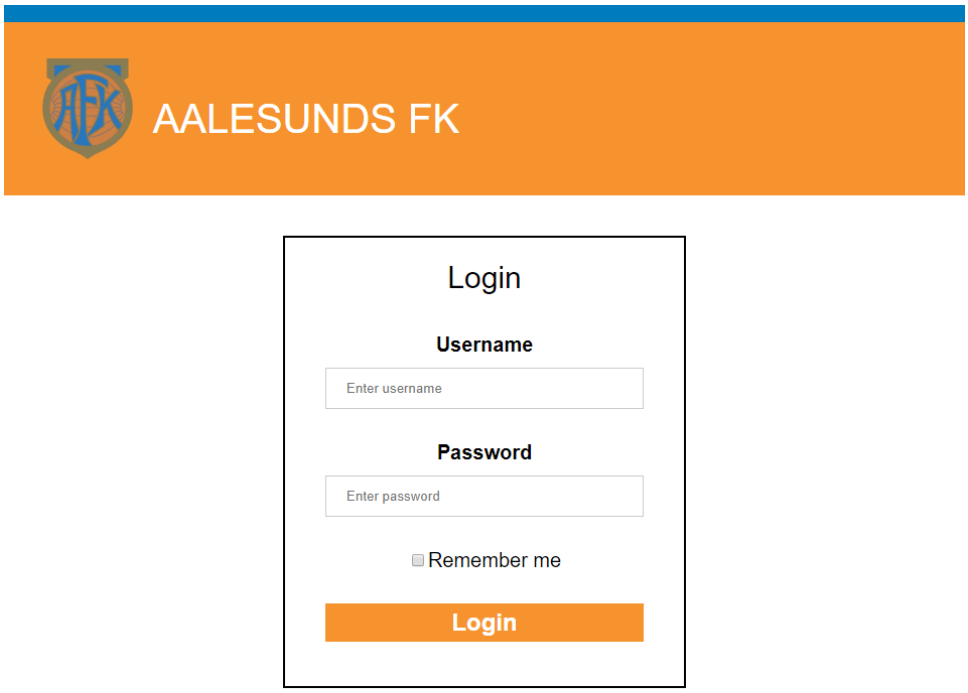
Følgende del av rapporten beskriver det grafiske brukergrensesnittet og inneholder bilder av nåværende versjon av applikasjonen.

Det grafiske grensesnittet er laget med tanke på at det skal være så enkelt og intuitivt for brukeren som mulig, samtidig som det stemmer overens med Aalesunds Fotballklubb sine fargekoder.

Det er innrettet slik at brukeren skal, så lett som mulig, forstå hvordan grensesnittet blir brukt. Dette er for å forhindre frustrasjon i brukeropplevelsen av brukergrensesnittet og minimerer behovet for opplæring i bruk av systemet

4.4.1 Innlogging

For å få tilgang til nettsiden må man først logge inn. Nettsiden skal kun brukes av ledelsen i AaFK og deres speidere. Innloggingsvinduet består av felt for innfylling av brukernavn og passord, og en knapp for innlogging. Dette er det eneste som er tilgjengelig for en bruker som ikke er pålogget. Derfor er brukergrensesnittet kun bestående av et innloggingsskjema og en tom navigasjonsbar, for at inntrykket av applikasjonen skal bli mer uniform.



Figur 10: Innloggingsvinduet.

I innloggingsformen vår bruker vi en handling tag (action) for å validere innloggingsdataen gjennom metoden «/validateLogin». «/» brukes for å gjøre spesifikke kall til servergrensesnittet gjennom en @app.route(«/funksjonsnavn», «methods=[‘metode’]») som vist i figur 13. Noen av metodene som kan brukes her er DELETE, POST og GET (slett, last opp og hent). Dette brukes for å begrense kallene som kan bli gjort til de forskjellige funksjonene. F.eks. både hente og laste opp informasjon

over samme nettverkskall.

```
login.html
1  {% extends 'index.html' %}
2
3
4  {% block login %}
5
6      <main>
7
8          <section class="loginForm">
9
10             <h2>Login</h2>
11             <form action="/validateLogin" method="post" novalidate...>
51         </section>
52     </main>
53
54 {% endblock %}
```

Figur 11: HTML-fil for innlogginssiden

"validateLogin" henter ut verdiene for brukernavn og passord fra skjemaet og kobler seg til databasen for å kjøre "sp_validateLogin" (en lagret prosedyre i MySQL).

```
@app.route('/validateLogin', methods=['POST'])
def validate_login():
    try:
        _username = request.form['username']
        _password = request.form['password']
        if regex.search(_username) is None and regex.search(_password) is None:
            # Connect to mySQL
            conn = mysql.connect()
            cursor = conn.cursor()
            cursor.callproc('sp_validateLogin', (_username,))
            data = cursor.fetchall()
```

Figur 12: Python fil for innlogging.

Denne prosedyren verifiserer om brukernavnet ligger i databasen. Prosedyren blir bare kjørt om det ikke er noen ulovlige tegn i verdien sendt inn for parameterne «username» og «password».

```
1  CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_validateLogin`(
2  IN p_username VARCHAR(20)
3  )
4  BEGIN
5  SELECT * FROM users WHERE user_username = p_username;
6  END
```

Figur 13: sp_validateLogin i MySQL Workbench

Hvis brukernavnet ligger i databasen vil "sp_validateLogin" hente den brukeren med brukernavn likt «p_username». «cur.fetchall()» henter listen av brukere fra denne spørringen.

Hvis en bruker er funnet i databasen vil funksjonen sjekke passordet, om brukernavnet ikke er funnet vil en feilmelding bli sendt.

```
    if len(data) > 0:
        if check_password_hash(str(data[0][4]), _password):
            session['user'] = data[0][2]
            return redirect(url_for('user_home'))
        else:
            return render_template('error.html', error='Wrong password. (change later)')
    else:
        return render_template('error.html', error='Wrong username and/or password')

except Exception as e:
    return render_template('error.html', error=str(e))
```

Figur 14: Verifisering om passordet oppgitt matcher brukerens passord i databasen.

Om brukernavnet er riktig må verdien av passordet sjekkes, det gjøres ved å generere en hashet verdien av innfyllingsfeltet for passord er lik den hashede verdien i databasen vil en sesjon for brukeren opprettes. Brukeren blir deretter sendt videre til velkomstsiden.

4.4.2 Etter innlogging

Etter innlogging vil brukeren bli sendt til velkomstsiden og kan begynne å navigere rundt nettsiden. Speidere og administratorer har forskjellig brukergrensesnitt, hovedsakelig for å begrense hva en speider skal kunne se og administrere.

Speidere skal kunne se og endre de spillerne de selv legger til databasen.

Administrator har tilgang til "skyggelag», de kan legge til og endre informasjon om speidere og agenter, og har tilgang til å se og endre informasjon om alle spillerne i databasen. De har også mulighet til å markere spillere som favoritter for raskere tilgang til de spillerne som er mest interessante på nåværende tidspunkt.

Figur 15: Innloggingsside ved vellykket innlogging for en speider.

Navigasjonsbaren er enkel og tydelig, og gjør det enkelt å navigere mellom sidene. Den gir også muligheten til å søke gjennom spillerne tilknyttet speideren i søkefeltet øverst til høyre.

Ved innlogging som administrator får brukeren tilgang til alle funksjoner i navigasjonsbaren.

Figur 16: Navigasjonsbar for administratorer.

En administrator må ha tilgang til flere funksjonskall enn det en speider har, og vil derfor ha flere navigasjonsknapper.

Figur 17: Navigasjonsbar for speidere.

Speideren har færre rettigheter og vil med det ha færre navigasjonsknapper. Dette gjøres for å minimere kompleksiteten i brukergrensesnittet og følger designprinsippet til Shneiderman's (Kap. 2.1.1) ved å skjule elementer uten effekt.

Innholdet av navigasjonsbaren blir bestemt ved å kalle til flask-funksjonen "getRole" ved bruk av AJAX i Javascript. Hvis returverdien av flask-funksjonen tilsvarer rollen Speider, kalles en funksjon som hindrer brukeren fra å få tilgang til navigasjonknappene som kun administratorer skal ha tilgang til.

```
$.ajax({
    url: 'getRole',
    type: 'GET',
    success: function (res) {
        if (res === "Scout") {
            ...
            .
```

Funksjonen «getRole» returnerer rollen til brukeren, enten i form av Administrator eller Speider. Først sjekker funksjonen om brukeren er pålogget systemet, hvis funksjonen returnerer ønsket svar, gjør den et funksjonskall til funksjonen «get_user_role()» i servergrensesnittet. Returverdien, som representerer rollen til den aktive brukeren, blir så returnert.

```
@app.route('/getRole')
def get_role():
    if session.get('user'):
        role = get_user_role(session.get('user'))
        return jsonify(role)
    else:
        return "No session"
```

Funksjonen «get_user_role()» tar inn et argument for brukernavn, som den her får via «session.get('user')». Siden applikasjonen ikke tillater flere brukere å ha samme navn, kan dette navnet med sikkerhet brukes for å hente ut riktig rolle fra databasen. Funksjonen skaper en kobling til databasen, før den henter ut og returnerer rollen til den entiteten i databasen som stemmer med navn-argumentet.

```
def get_user_role(user_name):
    conn = mysql.connect()
    cur = conn.cursor()

    name_string = swapper(user_name)
# Swapper konverterer listeellement til string.

if user_name:
        cur.execute('''SELECT user_role FROM users WHERE user_name LIKE
%s''', name_string)
        role = cur.fetchone()
        return role[0]
    else:
        "No session"
    cur.close()
```

4.4.3 Søkefunksjon

For å forenkle prosessen av å finne spillere er det viktig å kunne søke gjennom spillere basert på forskjellige kriterier.

Søkefeltet består av et felt hvor en String kan legges inn og en knapp for å starte søket. Dette kan også initieres ved å trykke Enter knappen på tastaturet mens brukeren skriver i feltet.

Figur 18: Søkefelt i HTML headeren

Deretter begrenser vi innholdet i søkebegrepet, slik at det skal være bedre beskyttet mot SQL-injeksjon.

I denne koden (figur 19) vil man kun søke etter navn, posisjon og beskrivelse av en spiller. For at det skal føles mer naturlig, brukes «%» på hver side av søkebegrepet. På denne måten blir søkebegrepet en del av navnet og det kreves ikke at søkebegrepet må være første del, siste del, eller hele navnet.

Etter søkebegrepet er blitt modifisert, blir det gjort et kall til «search.py» basert på brukerens rolle i sesjonen. En administrator skal kunne se alle spillerne i databasen, mens en speider skal kun se de spillerne de selv har lagt inn i databasen.

For å forhindre at en speider skal ha tilgang til informasjon den ikke selv har lagt inn, benytter vi oss av brukersesjonen for å få brukerens brukernavn og sende det gjennom et funksjonskall til «search.py».

```
@app.route('/search', methods=['get', 'POST'])
def search():
    if session.get('user'):
        _searchTerm = request.form['searchTerm']
        if regex.search(_searchTerm) is None:
            role = get_user_role(session.get('user'))
            _searchTerm = request.form['searchTerm']
            _searchTerm = "%" + _searchTerm + "%"
            if role == "Admin":
                player_id = search_player_name(_searchTerm)
                player_id += search_player_position(_searchTerm)
                player_id += search_player_description(_searchTerm)
                player_id = list(dict.fromkeys(player_id))
                if player_id:
                    f = get_player_info(player_id, True)
                    return jsonify(f)
                else:
                    return "player not found"

            if role == "Scout":
                g = search_scout_name(session.get('user'))
                player_id = search_my_players_name(_searchTerm, g)
                player_id += search_my_player_position(_searchTerm, g)
                player_id += search_my_player_description(_searchTerm, g)
                player_id = list(dict.fromkeys(player_id))
                if player_id:
                    f = get_player_info(player_id, True)
                    return jsonify(f)
                else:
                    return "player not found"
        else:
            return "Illegal signs in search term"
```

Figur 19: søke funksjonen i routes.py.

«search_player_name()» tar inn et argument i form av en String, og gjør et funksjonskall til databasen om å opprette en kobling basert på informasjonen fra «__init__.py». Deretter vil SQL spørringen bli utført for å få returnert en liste med spillerID'er.

```
def search_player_name(*args):
    conn = mysql.connect()
    cur = conn.cursor()
    player_ids = []
    cur.execute('''SELECT player_ID FROM basic_info WHERE player_name LIKE %s;''', (swapper(args),))
    for row in cur.fetchall():
        player_ids.append(row)
    return player_ids
```

Figur 20: Funksjonen "search_player_name()" i search.py.

Nesten all funksjonalitet i web-klienten er orientert rundt en representasjon av spillere. Derfor er det viktig å hele tiden hente den nyeste versjonen av spillerne, og oppdatere informasjonen ved navigering til en ny side. Dette gjøres ved at nesten alle sidene gjør minst et kall til search(). Det er viktig at koden for uthenting av informasjonen av spillerinformasjonen er så effektiv og modulær som mulig. Vi har segmentert koden slik

at det vil bli mer kohesjon. Dette blir gjort for å forenkle forståelsen av programmet og gjør at flere funksjoner kan bruke samme metode hver gang den logiske operasjonen trengs.

«get_player_info()» gjør et kall til «get_list_player()» med en liste spillere som parameter.

```
def get_player_info(args=None, table=None, edit=None):
    conn = mysql.connect()
    cur = conn.cursor()
    player_list = []
    if args is None:
        cur.execute("SELECT * FROM basic_info")
        player_list = get_list_players(cur.fetchall())
    else:
        for player in range(len(args)):
            cur.execute('''SELECT * FROM basic_info WHERE player_ID like %s''', (args[player]), )
            player_list += get_list_players(cur.fetchall())
    cur.close()
    if table:
        player_list = PlayerTable(player_list)
        if edit:
            return player_list.__html__()
    return player_list
```

Figur 21: Funksjonen «get_player_info()» i search.py

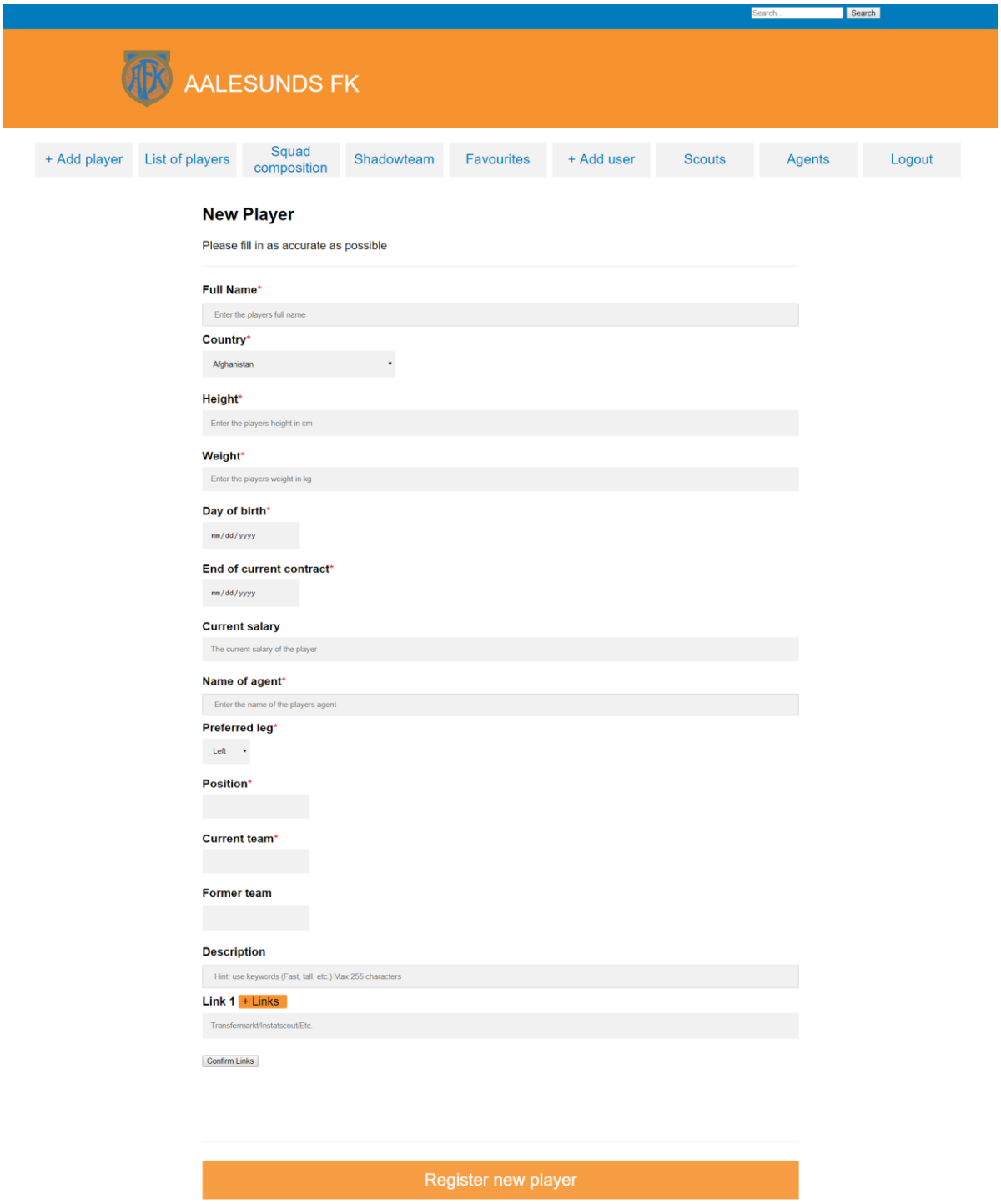
Basert på hvilke argumenter «get_player_info()» får, returnerer den en liste med spillerelementer.

```
def get_list_players(players):
    player_list = []
    for row in players:
        _name = row[1]
        _height = row[2]
        _weight = row[3]
        _birth_date = row[4]
        _nationality = row[5]
        _preferred_leg = row[6]
        _current_club = get_current_club(row[7])
        _former_club = get_former_club(row[7])
        _contract_expiry = get_contract(row[7])
        _salary = get_salary(row[7])
        _agent = get_agent(row[7])
        _scout = get_scout(row[7])
        _position = get_position(row[7])
        _description = get_description(row[0])
        player_list += [
            Player(_name, _height, _weight, _birth_date, _nationality, _preferred_leg, _current_club,
                   _former_club, _contract_expiry, _salary, _agent, _scout, _position, _description)]
    return player_list
```

Figur 22: "get_list_players()" funksjonen i search.py.

4.4.4 «Legg til spiller»-side

Denne siden lar brukere legge til spillere i databasen. Nye spillere kan enten være del av spillerstallen til AaFK eller spillere som speidere mener kan være interessante for AaFK å se nærmere på.

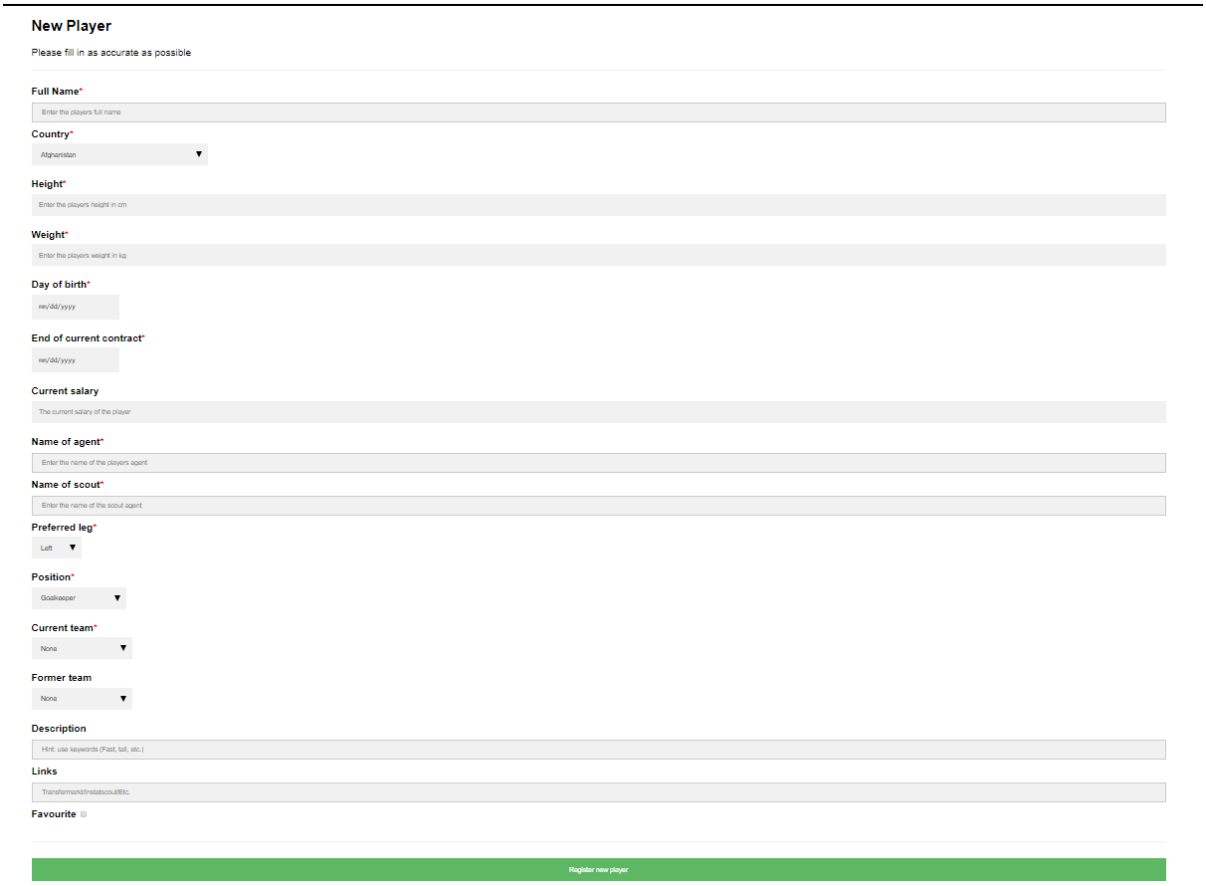


Figur 23: "Legg til spiller" siden.

Første utgave av siden hadde felt for å manuelt fylle inn speideren som er tilknyttet en spiller, begrensning på kun en lenke per spiller, en avkryssingsboks for å bestemme om spilleren er markert som favoritt og knappen for registrering av spiller var grønn. Siden hadde også nedtrekkingslister for feltene «posisjon», «nåværende lag» og «Forrige lag», dette begrenset brukeren til å kun velge mellom det posisjonene og lagene som var hardkodet inn i applikasjonen.

I den endelige utgaven av produktet har denne siden endret seg på følgende vis:

1. Speiderfeltet fylles ut automatisk ved å hente ut navnet til brukeren som registrer spilleren.
2. Avkryssingsboksen for favoritt er fjernet. Etter diskusjon innad i gruppen og med oppdragsgiver konkluderte vi med at administratorbrukere i stedet får muligheten til å endre på om en spiller er favoritt eller ikke på spillerprofilsiden. Alle nye spillere blir nå opprettet som ikke-favoritt.
3. Brukeren kan nå trykke på knappen «+ Lenker» for å få opp flere felt for innfylling av lenker. Slik kan brukeren legge til så mange lenker til en spiller som ønskelig.
4. Etter diskusjoner med veileder ble vi enige om å begrense fargevalg til AaFK sine fargetoner. Derfor er fargen på knappen på siden, og flere slike elementer brukt på siden endret.
5. Feltene «posisjon», «nåværende lag» og «forrige lag» bruker nå ett innfyllingsfelt i stedet for en nedtrekkingsliste. Det nye innfyllingsfeltet gir fortsatt en nedtrekkingsliste som gir forslag om klubber og posisjoner som er brukt før, for å minimere duplikatnavn (i.e. AaFK og Aalesunds FK). I tillegg gir det mulighet for å fylle inn andre verdier enn de fastsatte. Dette gjorde vi etter samtaler med oppdragsgiver, som ønsket at systemet skulle være mer tilpasningsdyktig på disse områdene i tilfelle klubben endrer lagformasjon eller utvider speidernettverket sitt.



Figur 24: Tidlig utgave av "Legg til spiller" siden.

For å legge til mer enn en lenke per spiller kan brukeren trykke på "+Links"-knappen for å åpne flere felt for innfylling av lenker.

Figur 25: Linkseksjonen av "legg til spiller"-siden med tre felt.

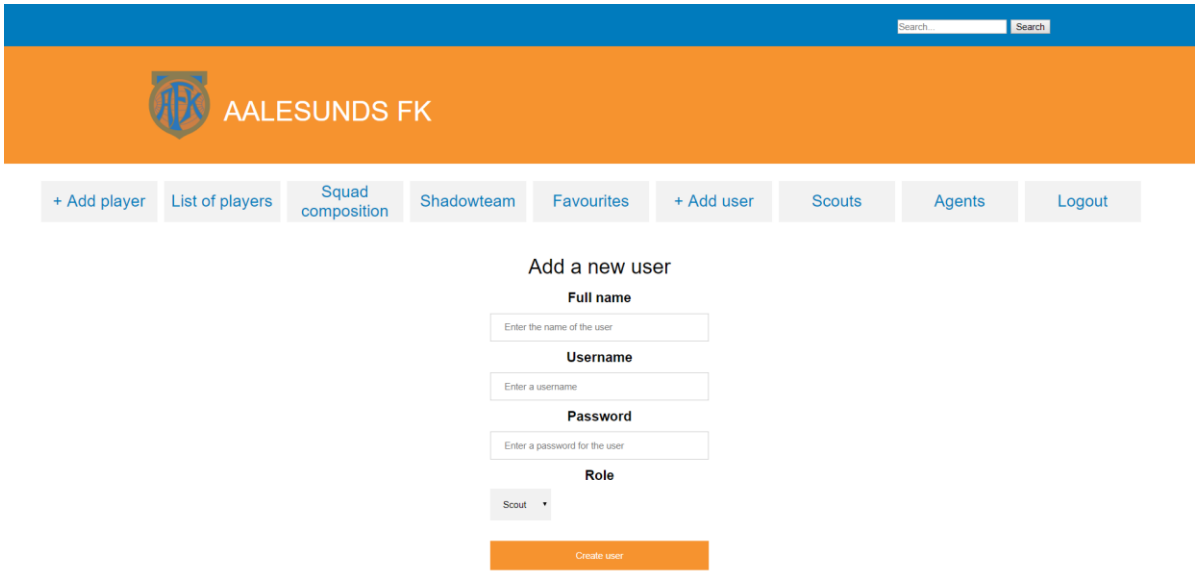
Når brukeren trykker på «registrer ny spiller»-knappen, kalles Flask funksjonen "add_player()" i routes.py. Denne funksjonen henter informasjon fra siden ved bruk av skjemaer. Informasjonen hentet blir sjekket for tegn som kan være skadelige og føre til SQL-injeksjoner. Hvis ingen av feltene inneholder skadelige tegn, skapes en forbindelse til databasen, og det kalles på den lagrede prosedyren "sp_createPlayer". Den lagrede metoden tar inn argumentene og oppretter en ny spillerentitet, samt alle knyttede entiteter.

```
@app.route('/addPlayer', methods=['POST'])
def add_player():
    # Reads the posted values from the UI
    _player_name = request.form['name']
    _country = request.form['country']
    _height = request.form['height']
        # Dette mønsteret blir brukt på flere attributter.

_scout = session.get('user')
    # Validate the received values
    if regex.search(_player_name) is None and regex.search(_country) is None and
        regex.search(_height) is None and regex.search(_weight) is None
        # Dette mønsteret blir brukt på flere attributter.

          conn = mysql.connect()
        cursor = conn.cursor()
        cursor.callproc('sp_createPlayer',
                    (_player_name, _country, _height, _weight,
                                _birth, _contract, _salary, _agent
                                ,_scout, _leg, _position, _team, _former_team,
                                _description, _links, _date_published))
        data = cursor.fetchall()
        if len(data) is 0:
            conn.commit()
            return json.dumps({'message': 'Player created
                                                successfully.'})
        else:
            return json.dumps({'error': str(data[0])})
```

Figur 26: Fuksjonen "add_player()" i routes.py.

4.4.5 «Legg til bruker»-side

Dette er en administrasjonsside for AaFK til å legge til brukere som skal ha tilgang til siden. Administratorer kan her fylle inn navn, brukernavn, passord og rolle for den nye brukeren.



Figur 27: "Legg til bruker"-siden.

Funksjonelt kaller «Lag bruker»-knappen på en funksjon som er relativt lik "add_player()" men vil også kryptere noe av dataen slik at en hashet verdi lagres istedenfor passordet i klartekst. Det å lagre passord i klartekst er dårlig praksis med tanke på bibliotekangrep. Derfor brukes Werkzeug sin «generate_password_hash» for å generere en hashet versjon av brukerens inntasting av passordet. Hovedgrunnen til dette er for å garantere at brukerens passord ikke skal bli funnet om et eventuelt glipp i applikasjonens sikkerhet gir tilgang til databasens lagrede verdier av passord.

Etter passordet er hashet kalles den lagrede prosedyren "sp_createUser" som setter opp en ny brukerentitet i databasen. Den lager også en entitet i speider- og ansatt tabellene hvis den nye brukeren har rollen «speider». Hvis det eksisterer en ansattentitet med samme navn som den nye brukeren i databasen knyttes de sammen i stedet for å lage nye entiteter.

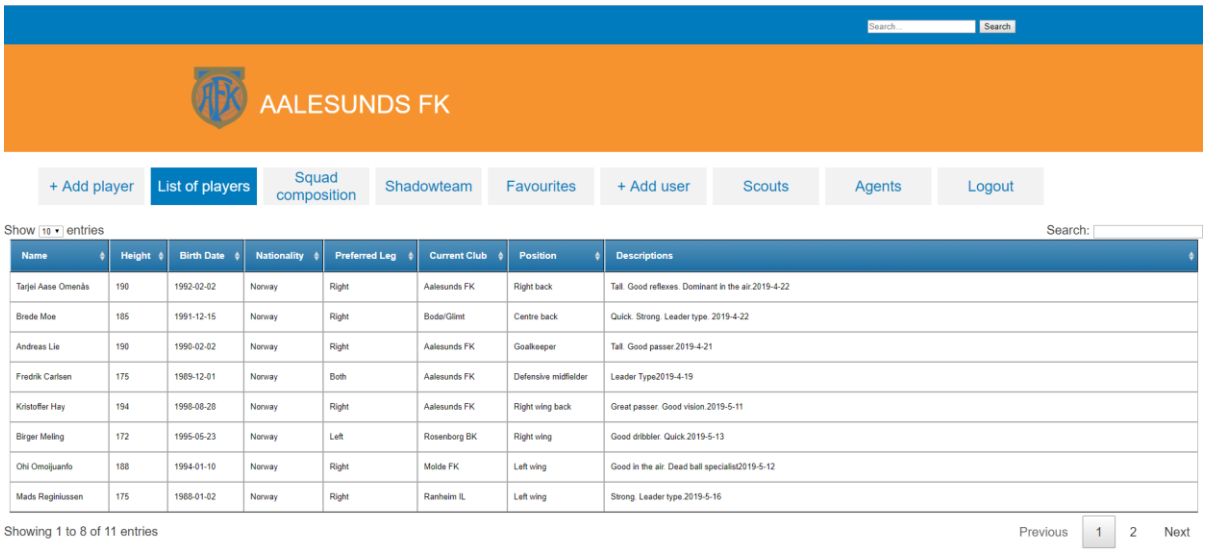
```
CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_createUser`(
    IN p_name VARCHAR(45),
    IN p_username VARCHAR(45),
    IN p_password VARCHAR(255),
    IN p_role VARCHAR(20)
)
BEGIN
IF (SELECT EXISTS (SELECT 1 FROM users WHERE user_username = p_username) ) THEN
            SELECT 'Username already exist';
ELSE
INSERT INTO users (user_name, user_username, user_password, user_role)
       VALUES (p_name, p_username, p_password, p_role);

SELECT user_ID
INTO @us_id
FROM users
```

```
WHERE user_name = p_name;
END IF;

IF (p_role = "Scout") THEN
IF ( SELECT EXISTS ( SELECT 1 FROM employee_info WHERE employee_name = p_name ) ) THEN
SELECT info_ID
INTO @e_ID
FROM employee_info
WHERE employee_name = p_name;

ELSE
INSERT INTO employee_info (employee_name) VALUE (p_name);
SELECT info_ID
INTO @e_ID
FROMemployee_info
WHERE employee_name = p_name;
END IF;

INSERT INTO scout (user_scout_id, scout_employee_id) VALUES (@us_ID, @e_ID);
END IF;
END
```

Figur 28: Den lagrede prosedyren sp_createUser.

4.4.6 Spillerlistesiden

Denne siden gir en liste over alle spillere i databasen for AaFK. For speidere vises kun spillere de selv har lagt til databasen. I utgangspunktet brukte vi en egen Javascript-funksjon for å sortere enkeltkolonner. Oppdragsgiver kom senere i utviklingsprosessen med ønske om å kunne søke med to forskjellige parametere. Derfor bruker tabellen nå biblioteket DataTable som bidrar med en del funksjonalitet som forbedrer brukervennligheten på flere måter. Fra DataTable har vi implementert metoder som sørger for sortering, inndeling i sider og en ekstra søkefunksjon, som gjør at brukeren nå kan søke med to forskjellige parametere.



Figur 29: Spillerlistesiden

4.4.7 Favorittsiden

Favoritter er en side tilgjengelig for AaFK til å gruppere spillere som er av stor interesse for rask tilgang. Siden er kun tilgjengelig for administratorbrukere. På denne siden kan en administrator se alle spillere som er lagret som favoritter i databasen. Siden har en horisontal liste med spillerelementer. Her får en administrator se alle spillere som er markert favoritter. Alle elementene i denne listen linker til tilsvarende spillerprofil. Under listen er den en visuell representasjon av alle favorittene i sin posisjon på banen. Denne visningen inneholder alle de forskjellige posisjonene i systemet.

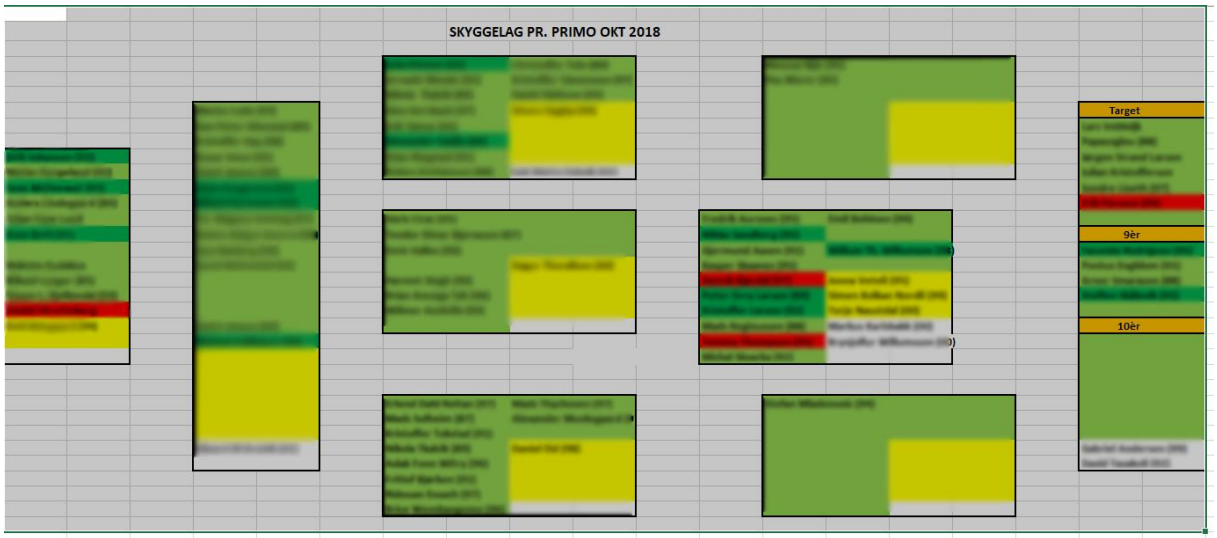
Figur 30: Favorittsiden

4.4.8 «Skyggelag»-side

Denne siden viser er grafisk grensesnitt av en fotballbane som viser alle favoritter som kan fylle en rolle i AaFK sin nåværende formasjon. Favorittspillere som spiller i posisjoner som ikke er del av AaFK sin nåværende tropp blir ikke vist på denne siden. Hensikten med siden er å gjøre det enklere for ledelsen i klubben å identifisere spillere som kan fylle roller i troppen hvor det er mangel på alternativer i eksisterende stall.

Figur 31: "Skyggelag"-siden

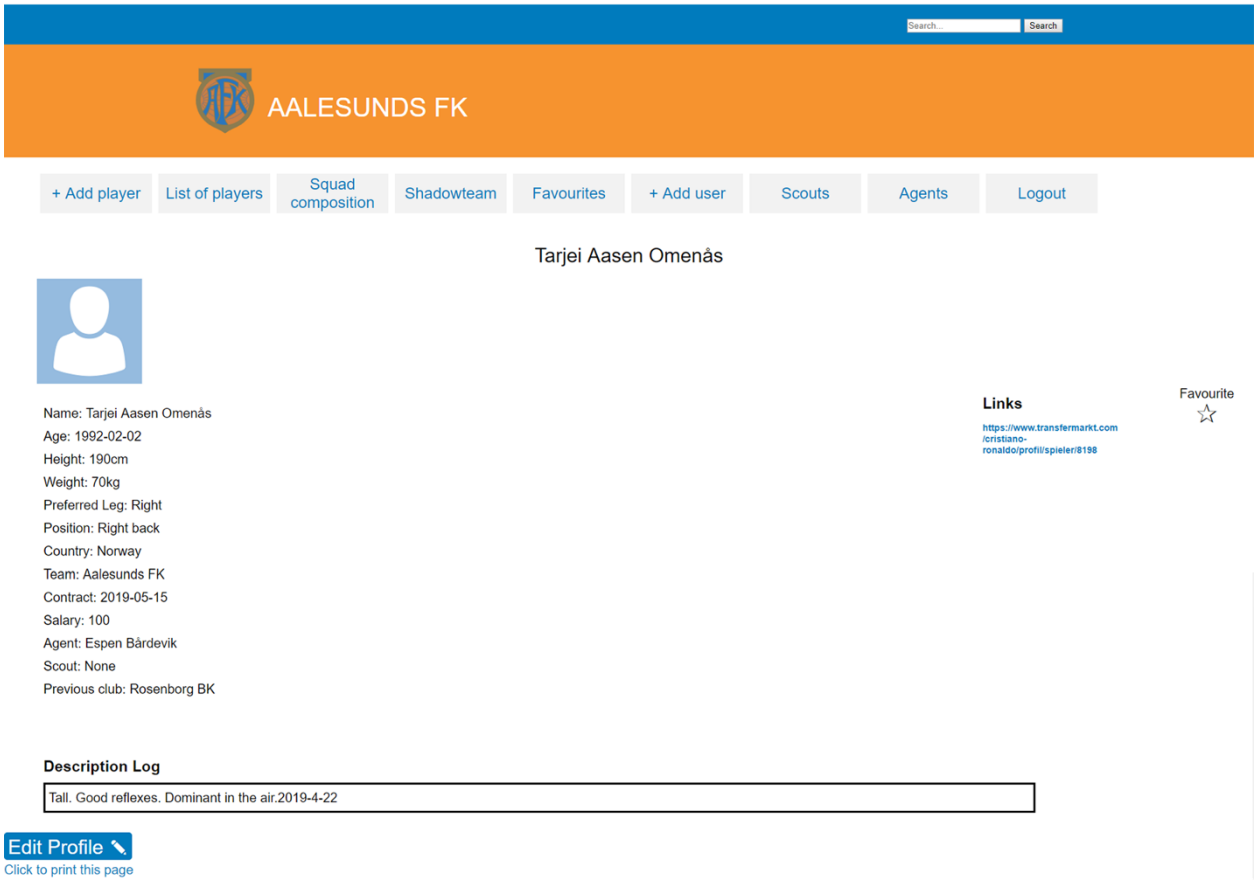
Denne siden (figur 31) er skapt for å erstatte klubbens nåværende system med samme funksjon. Fram til nå har skyggelag blitt satt opp og oppdatert manuelt i Microsoft Excel *(Figur 32)*. Hvis en spiller skal legges til eller fjernes fra skyggelaget i vår applikasjon, kan en administrator endre på favorittstatusen til en spiller fra dens profilside. Skyggelaget blir da automatisk oppdatert.



Figur 32: Nåværende system for skyggelag. Spillerinformasjon fra bildet er gjemt slik at potensielt sensitiv informasjon ikke skal være synlig.

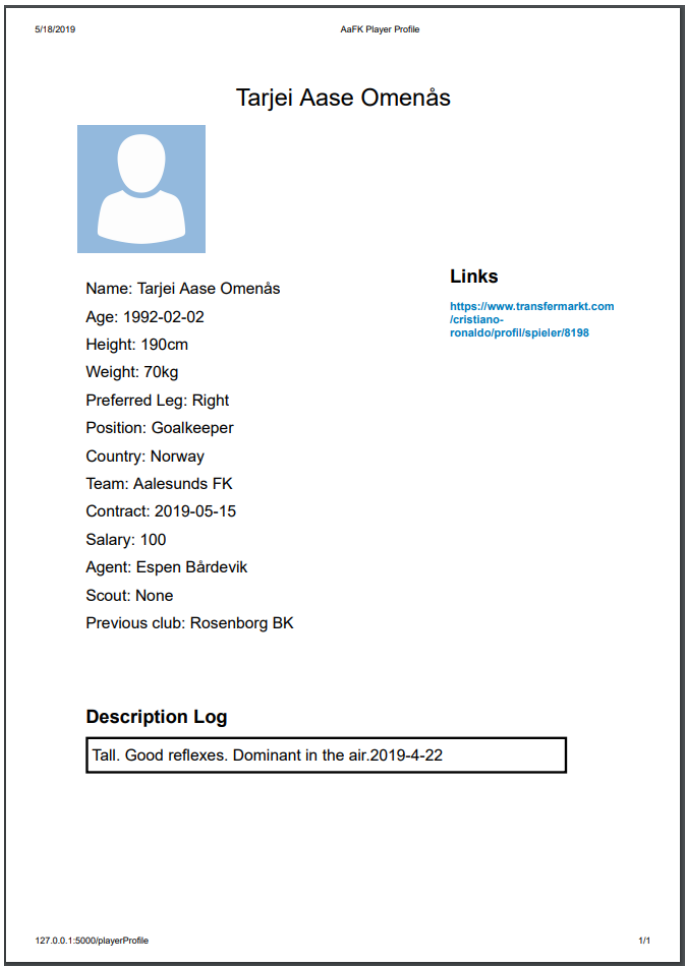
4.4.9 Spillerprofilside

Denne siden viser profilen til en spiller i databasen. Det er mulig å endre/oppdatere informasjon om en spiller og legge til ekstra kommentarer om spilleren. Siden har en liste med detaljert informasjon om en spiller, et felt for lenker assosiert til spilleren, et felt som viser en beskrivelse av spilleren, en boks som viser om spilleren er registrert som favoritt eller ikke og knapper for redigering av informasjon og utskrivning av siden. Hvis favorittknappen blir trykt, endres spillerens status til "favoritt" eller "ikke favoritt" i databasen. Boksen er gul hvis spilleren er favoritt, ellers er den gjennomsiktig, som i bildet under.



Figur 33: Spillerprofilsiden

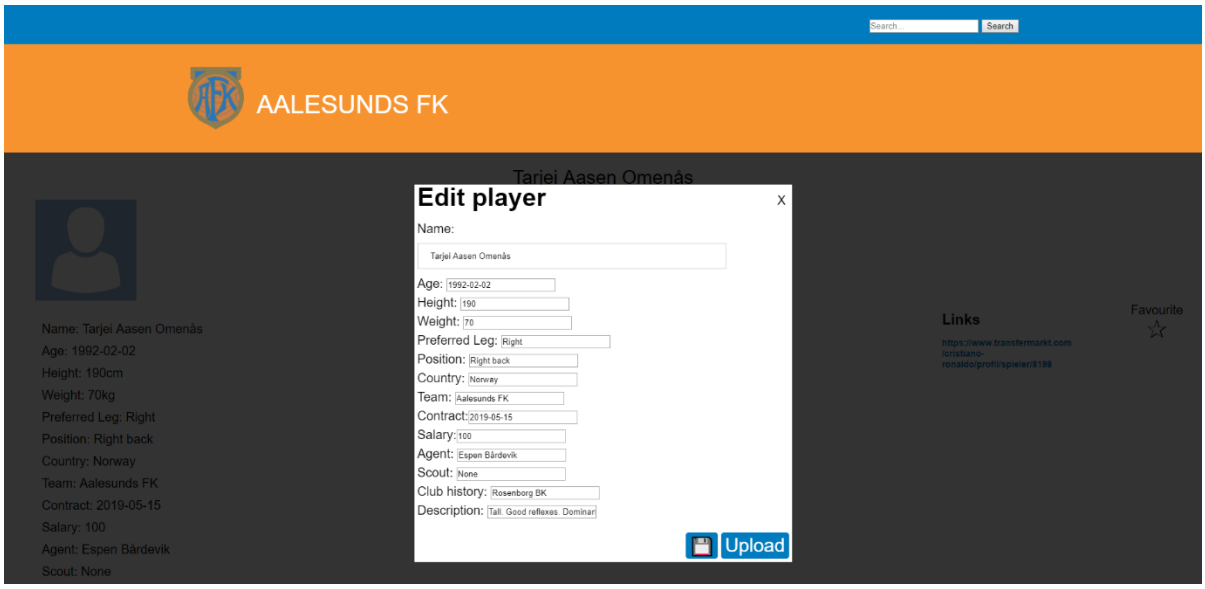
Ved utskrift av siden blir kun spillerinformasjonen tatt med. Slik kan utskriften brukes til å vise spilleren videre for vurdering angående eventuelt spiller-kjøp.



Figur 34: Utskrift av spillerprofil.

4.4.9.1 *Redigering av spillerprofil*

Hvis brukeren ønsker å endre på informasjonen om en spiller kan han/hun trykke på "Rediger profil"-knappen. Da kommer det opp et vindu med innfyllingsfelt som holder på nåværende informasjon. Brukeren kan da endre på de feltene som er ønskelig, trykke lagre og last opp. Da vil all informasjon som er endret bli oppdatert i databasen. Hvis knappen for lagring ikke blir trykket før opplastingsknappen, får brukeren en feilmelding i visningen som sier at den må trykkes før opplastingsknappen og ingen endringer bli gjort.



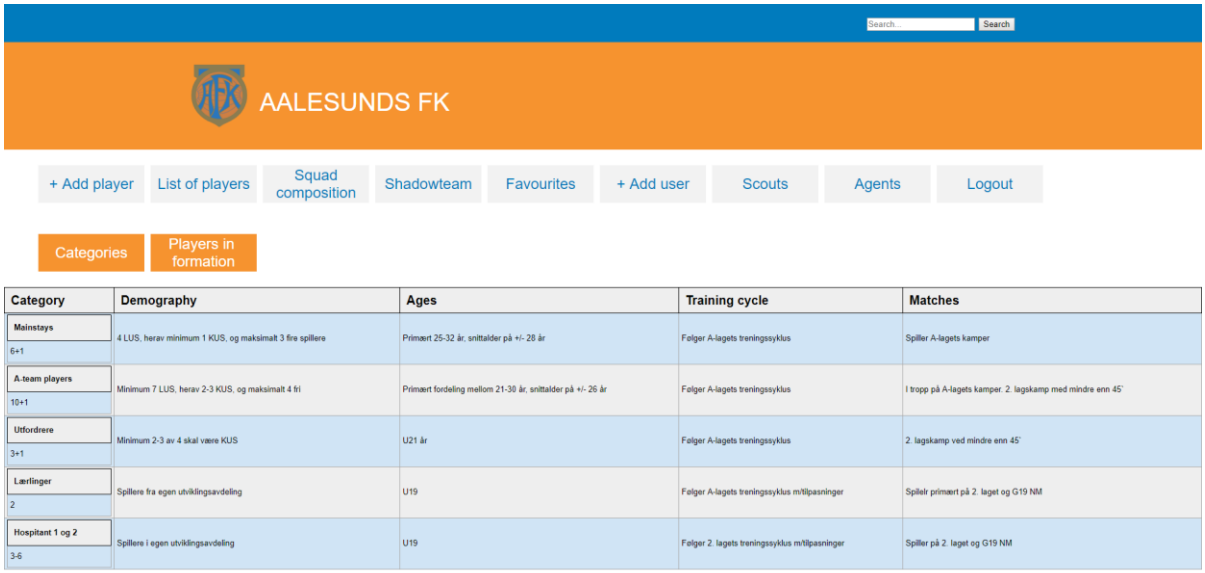
Figur 35: Felt for endring av spillerinformasjon.

4.4.10 Stallkomposisjonsiden

Stallkomposisjonssiden brukes til å holde system på klubbens nåværende spillerstall. Siden har to forskjellige visninger: Kategorisiden og stallvisualiseringssiden. Kun brukere med administratorrettigheter har tilgang til denne siden.

4.4.10.1 Kategorisiden

Kategorisiden har ingen funksjonalitet. Den er kun til for å hjelpe administratorer med å huske hvordan spillerstallen bør være sammensatt, ut fra klubbens egne regler og verdier.



Figur 36: Kategorisiden

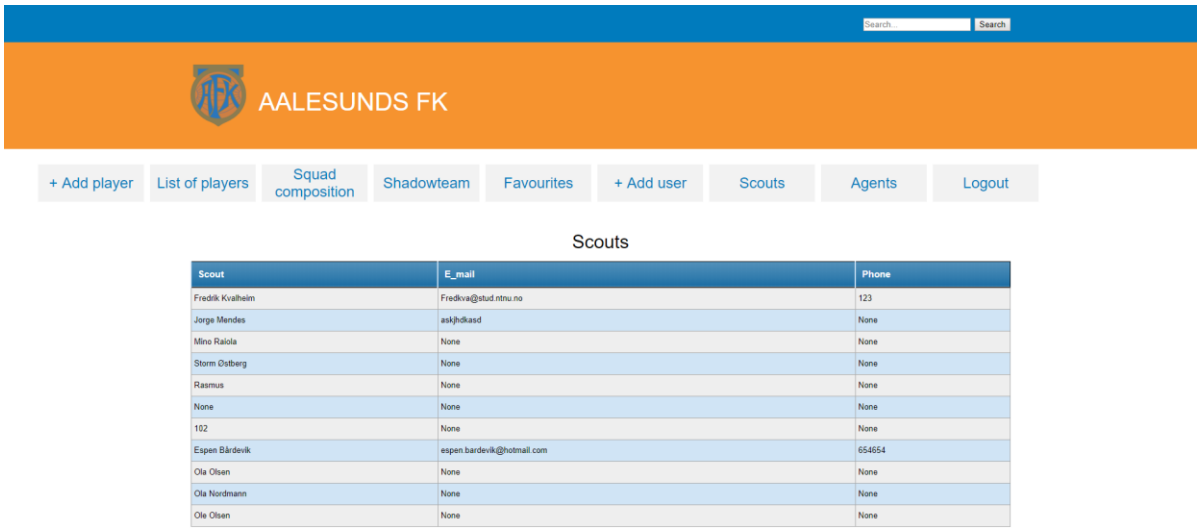
4.4.10.2 Stallvisualiseringssiden

Stallvisualiseringssiden har lik funksjon som skyggelagsiden. Forskjellen er at denne siden viser spillere registrert i klubben ved angjeldende tidspunkt. Etter samtaler med oppdragsgiver har vi kommet fram til at dette er posisjonene som brukes i klubbens nåværende system. Siden viser alle spillere i klubben som kan fylle de forskjellige rollene og brukes til å identifisere problemområder og mangler.

Figur 37: Side for visualisering av spillere i klubbens standardformasjon.

4.4.11 Ansattsider

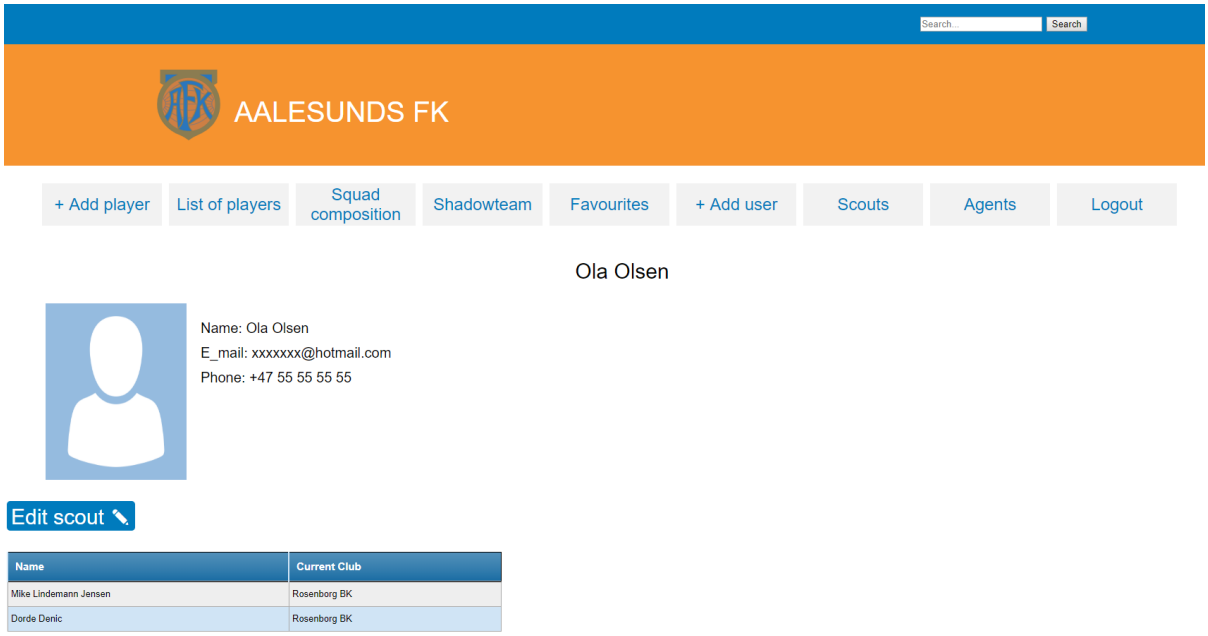
Ansattsidene "Speidere" og "Agenter" er administrasjonssider som viser lister over alle speidere og agenter registrert i databasen. Listene holder på kontaktinformasjon om hver enkelt ansatt. Hvert listeelement er en lenke til profilsiden til tilhørende ansatt.



Figur 38: Speiderlistesiden.

4.4.12 Ansattprofilsider

På ansattprofilsidene vises en speider eller agent med deres kontaktinformasjon. Kontaktinformasjonen kan endres på samme vis som spillerinformasjon på spillerprofilsiden. En ansattprofil har også en liste med alle spillere som er tilknyttet den ansatte.



Figur 39: Speiderprofilsiden.

5 Drøfting

5.1 Det grafiske brukergrensesnittet

Vi fikk tidlig vite at det grafiske designet på nettsiden ikke var av høyest prioritet, men at det måtte bære preg på at det var en nettside for Aalesund fotballklubb.

Som en hvilken som helst nettside burde den være intuitivt og så simpel som mulig.

For grensesnittet tok vi inspirasjon fra Aalesunds fotballklubb sin egen nettside, aafk.no.

5.2 Resultatet

Resultatet er en applikasjon som vi kan si oss godt fornøyd med, og som vi har fått inntrykk av at oppdragsgiver også er veldig fornøyd med. Applikasjonen i nåværende utgave automatiserer mange av oppgavene som måtte gjøres manuelt av oppdragsgiver tidligere. Vi har noen ideer til løsninger for videre utvikling vi kunne tenke oss å iverksette, men som har blitt nedprioritert på grunn av fokus på rapporten.

Prosjektet har oppfylt de vesentlige kravene for funksjonalitet og applikasjonen har en struktur som vi mener skal være relativt enkel å videreutvikle, hvis det er ønsket. Vi mener applikasjonen oppfyller sitt viktigste mål, å redusere arbeidsmengden for håndtering av spillerlogistikk så mye som mulig.

5.2.1 Forside

Originalt hadde vi tenkt oss en forside med kun banneret til Aalesunds FK, og en oversiktsside med knapper for å navigere til andre sider. På de andre sidene måtte en tilbakeknapp trykkes for navigering tilbake til fremsiden.

Vi endte med å beholde banneret, men å heller lage en navigasjonsbar som vises på hver enkelt side. På denne måten slipper bruker å måtte navigere tilbake til forsiden for så å gå til neste side.

Knappene er enkle og utformet i den stilen som blir brukt på aafk.no.

5.2.2 Liste av spillere

Denne siden var opprinnelig laget for å teste ut hvordan vi kunne endre på innholdet i en side og legge inn informasjon fra databasen. Vi endte opp med å bruke siden for å vise spillere som er lagt til i databasen, og utviklet videre på den slik at den kunne brukes i sluttresultatet.

5.2.3 Spillerprofil

Denne visningen er den som har krevd klart mest diskusjon med oppdragsgiver. Til slutt ble vi enige om hvilken informasjon som er vesentlig for klubben å ha i en spillerprofil. Denne siden var spesielt viktig å representere informasjon best mulig på, da oppdragsgiver ønsket å kunne bruke utskrift av siden til å presentere de mest interessante spillerne til andre, herunder styret i klubben. Det var også av stor betydning at spillerprofiler skulle være enkle å redigere, noe vi synes vi har funnet en god løsning på med redigeringsfeltet.

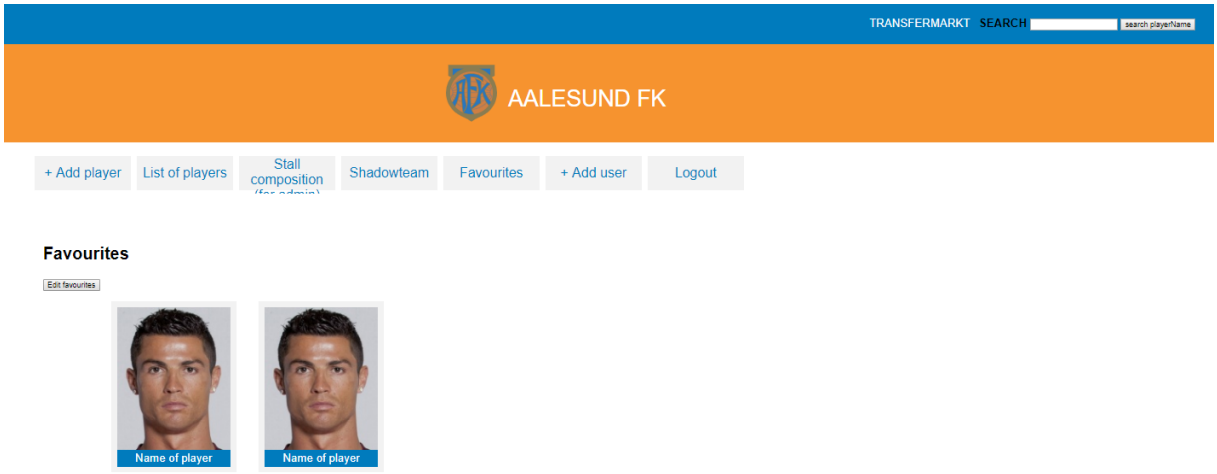
5.2.4 Stalloppsett

Dette er nok den siden vi ser mest potensiale til med videre arbeid. Siden viser nå fordeling på spillere i klubben etter startposisjon på banen, og kategorivisningen som kan brukes som retningslinjer for stallsammensetning. Visningen for fordeling av spillere automatiserer en oppgave som tidligere har krevd manuell oppdatering og vil derfor være til nytte for oppdragsgiver. Vi har flere ideer for videreutvikling av denne delen av applikasjonen som vi utforsker videre i Veien videre (Kap. 5.4).

5.2.5 Favoritter

Oppdragsgiver ønsket en metode for å kategorisere de mest interessante alternativene blant spillerne de speider på. Måten vi løste dette på var med å gi spillere en favorittattributt og å lage favoritt- og skyggelagsiden. Favorittattributten kunne i utgangspunktet bli satt av alle brukere, men etter diskusjoner med oppdragsgiver ble det klart at kun administratorer skulle kunne bestemme dette. Det ble tydelig at hvis alle brukere kunne sette og endre denne attributten, selv om det var kun for sine egenregistrerte spillere, ville det skape unødvendig rot i systemet hos administratorene. Favoritter kan derfor bare endres på av administratorer i den endelige utgaven.

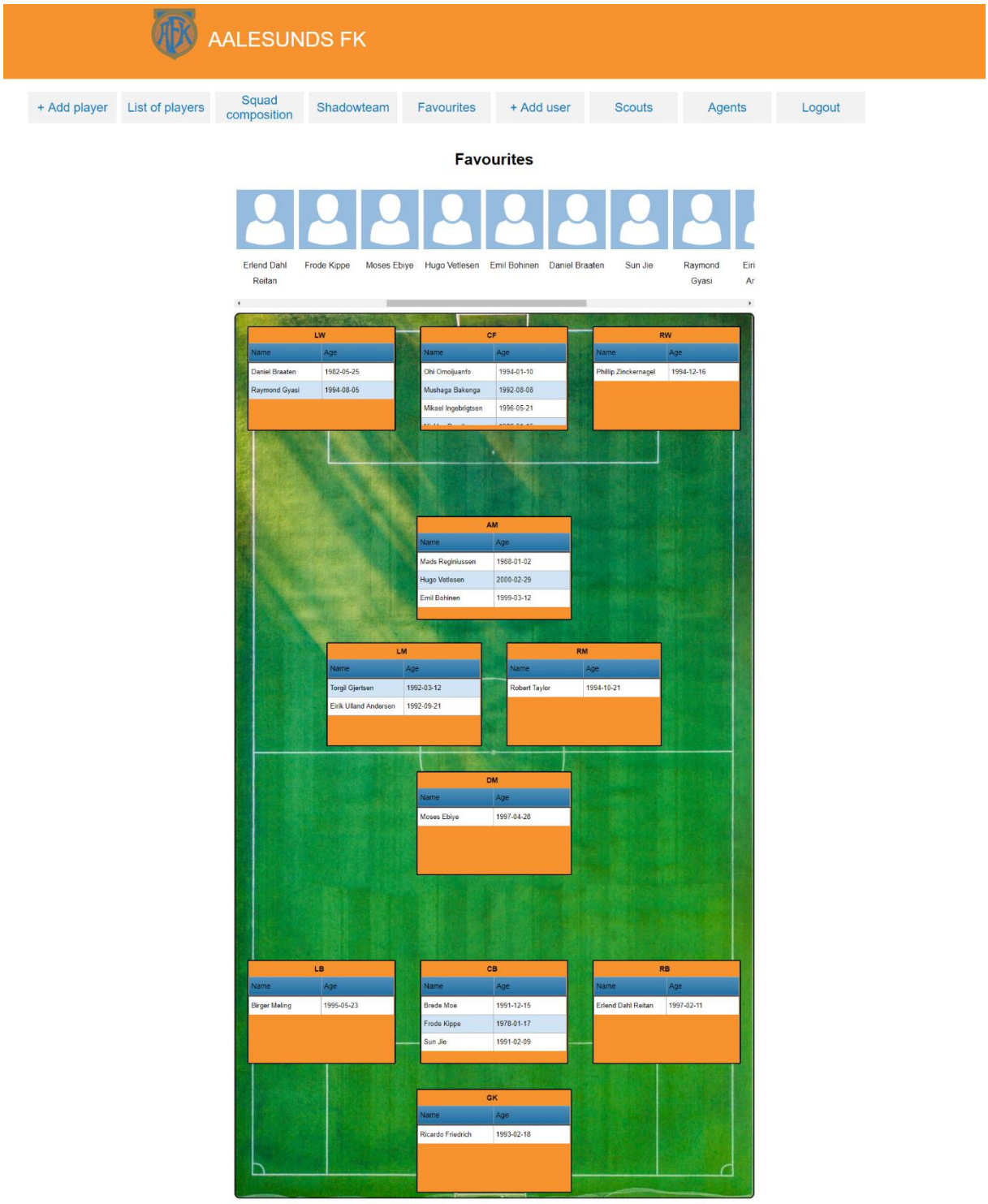
Grafisk har vi endret en del på favorittsiden. I utgangspunktet satte vi opp en struktur som skulle vise et galleri med spillerprofiler.



Figur 40: Tidlig utgave av favorittsiden.

Denne måten skulle vise alle favoritter som elementer med navn og bilde, og kun navn og bilde. Denne metoden var tenkt som en løsning til både favoritter og «skyggelag»-siden, da vi nedprioriterte disse funksjonaliteten til all kjernefunksjonalitet var implementert. Problemet var at det ikke var noen visualisering for spillernes plassering på banen. Etter all kjernefunksjonalitet var på plass begynte vi å se på forskjellige løsninger for å visualisere favoritter og skyggelag best mulig. Første utgave innebar å sette opp disse elementene med bilder og tekst i designerte områder basert på posisjon på banen. Dette endte opp med å bli alt for uryddig og lite intuitivt.

Det endelige produktet har derfor en struktur som vi mener har det beste av begge metodene. Spillerobjekter med bilde og navn er satt i en horisontal liste øverst på siden og visualiseringen bruker spillerlister med mer komprimert informasjon. Vi valgte også å legge til en gressmatte som bakgrunn, for å gjøre sidene mer estetisk tiltalende og intuitive.



Figur 41: Endelig utgave av favorittsiden.

5.2.6 «Skyggelag»

Dette var en ønsket funksjon som ikke var prioritert som en kjernefunksjon. Det vi har laget er en visuell representasjon av et skyggelag som automatiserer innfylling av spillerdata. Skyggelaget hjelper oppdragsgiver å holde en oppdatert visning på de mest interessante spillerne som er identifisert til å kunne fylle en rolle i klubbens startoppstilling. Vi har noen ideer for hvordan vi ville forbedret siden ytterligere, men mener at nåværende utgave er tilstrekkelig og forenkler en arbeidsoppgave for oppdragsgiver.

5.2.7 Legg til spiller

Å kunne legge til spillere i databasen fra applikasjonen var den aller viktigste kjernefunksjonaliteten til dette prosjektet. Måten vi har gjort det på er ved å lage en egen visning for registrering av spillere som holder på felt hvor all informasjon koblet mot en spiller kan bestemmes. Gjennom utviklingsprosessen har det vært noen endringer i krav om hvilken informasjon som skal kunne lagres og hvordan den skal håndteres. Informasjonen som nå lagres er bestemt etter samtaler med oppdragsgiver, og dette området av kravet er nå oppfylt.

5.2.8 Speidere og agenter

Ett av de nyere ønskene til oppdragsgiver var å kunne se og endre informasjon om speidere og agenter. Det var hovedsakelig et ønske om å kunne se og endre kontaktinformasjonen til de ansatte. Dette er funksjonalitet som kun administratorer har. Ved å trykke seg inn på speider- eller agentsiden får en administrator tilgang til en liste med speidere eller agenter. Listen viser også kontaktinfo til alle de ansatte.

For å endre kontaktinformasjonen kan en administrator trykke på en ansatt i listen. Da blir brukeren navigert til profilen til denne ansatte. Derifra kan kontaktinformasjonen endres. Her har vi allerede oppfylt kravet til oppdragsgiver. Vi bestemte oss likevel for å legge til en liste av spillere som er knyttet til denne ansatte, noe som oppdragsgiver virket fornøyd med. Denne ekstra funksjonaliteten ble derfor beholdt.

5.2.9 Søkefunksjon

For å gi rask tilgang til søk etter en spiller, er søkefunksjonen implementert i headeren av nettsiden, og den virker uavhengig av hvor i nettstedet brukeren er.

Søket returnerer relevant informasjon tilbake i en liste og det tillater sortering etter flere kriterier. Søket øker brukervennligheten ved å tillate bruk av delvis String (søket returnerer elementer som inneholder begrepet som er sendt inn). Dette gjør søket mer naturlig, og oppfølger kravet om å forenkle, automatisere og effektivisere prosessen med å finne frem til spillerinformasjon.

5.3 Utviklingsmetodikk

Opprinnelig hadde vi tenkt oss å bruke utviklingsmetodikken SCRUM under Agile utviklingsmetodikker siden det er en metode vi har erfaring fra tidligere fag og med gode resultat. Med SCRUM settes det opp en overordnet liste med oppgaver som skal bli gjennomført med prosjektet. Disse oppgavene blir delt inn i flere mindre håndterbare oppgaver som skal kunne fullføres i løpet av timer. Der er det også et poengsystem for oppgavene som forskjellige utvikler team løser på forskjellige måter, enten ut ifra tid oppgaven tar eller ut ifra hvor kompleks oppgaven er.

Vi fant ut etter et par intervaller med SCRUM at det var naturlig god dialog i gruppen, men at vi følte det var vanskelig å sette korte delmål. Dette ble også spesielt vanskelig når regelmessige møter med oppdragsgiver ble vanskeligere å realisere. Derfor byttet vi utviklingsmetodikk til en plandreven utvikling. Dette tillatte mer langsiktig planlegging med klare definerte delmål

Møtene med veileder ble holdt hver andre uke og vi prøvde å holde møte med oppdragsgiver på samme uke. Det viste seg ikke alltid å være like lett. Prosjektet løp ut over oppstart på fotballsesongen i Norge, noe som gjorde det vanskeligere for oppdragsgiver å holde jevnlige møter. Vi løste dette ved å holde kontakten gjennom e-post når det var nødvendig.

5.4 Veien videre

Interaktive diagram

Det vi mener bør bli gjort fremover med applikasjonen, er å utnytte mer den informasjonen som databasen nå holder på; for eksempel å legge til en funksjon som regner ut gjennomsnitt av forskjellige verdier som f.eks. høyde, alder eller lønn. All dataen vil allerede være tilgjengelig i databasen.

Skyggelag

Forbedre skyggelaget utseendemessig, gjøre funksjonen interaktiv; som å dra spillere til forskjellige posisjoner. Bruke bilder av spillerne i stedet for bare navn.

Visualisering av lønn -og aldersfordeling i troppen

Vi hadde også håpt å få tid til å iverksette sider som viste mer detaljert info om stallkomposisjonen, for eksempel visualisering av lønnsfordeling og aldersfordeling. Dette er funksjonalitet som ikke var del av de opprinnelige ønskene til oppdragsgiver. Gjennom prosessen har dette kommet fram som funksjoner som ville hjulpet oppdragsgiver med å effektivisere arbeidet med spillerlogistikkhåndtering. På grunn av begrenset tid, ble planene om utvikling av disse funksjonalitetene ikke iverksatt.

5.5 Hva har vi lært?

Med dette prosjektet har vi fått bruke den kunnskapen som vi har tilegnet oss gjennom våre tre år ved skolen og kombinert det.

Gjennom skolen har vi fått erfaring med å bruke HTML, CSS, JavaScript osv. i ulik grad. I prosjektet fikk vi testet ut erfaringen vår og brukt det til et større prosjekt som krevde en forståelse av hvordan de fungerer med hverandre.

Enkelte medlemmer i gruppen har ingen tidligere erfaring med Python, men nødvendig kunnskap ble opparbeidet gjennom arbeidet i prosjektet. Vi lærte om de mange nyttige funksjoner det tilbydde og følte at vi fikk gode resultat av valget vårt.

Prosjektet er også første virkelige mulighet vi har hatt til å jobbe som en gruppe med ett større prosjekt og en ordentlig oppdragsgiver. Vi har gjennom prosjektet fått større forståelse for hvordan arbeid og ansvar bør fordeles mellom utviklere. Møter med oppdragsgiver og planlegging derfra var i utgangspunktet mer utfordrende enn forventet. Vi har fått bedre innsyn i hvordan man, sammen med oppdragsgiver, bestemmer krav og arbeidsoppgaver framover.

6 Konklusjon

Gjennom prosjektet har gruppen laget et produkt som gir Aalesunds FK en måte å samle all informasjon de trenger om spillere på en felles applikasjon som er tilgjengelig gjennom en nettside.

Vi har laget funksjoner for å legge til og endre spillere og brukere (som speider og administrator). Mulighet for å søke gjennom spillere i databasen ved bruk av navn, posisjoner eller annen informasjon som er lagret i databasen. Den visuelle representasjonen av spillere i sine posisjoner på banen var en krevende funksjonalitet som vi hadde tvil om vi greide å implementere innen tidsrammen. Vi er derfor svært fornøyd med å ha utviklet en løsning for denne delen av oppgaven som både vi og oppdragsgiver er fornøyd med.

Selv om vi har valgt å følge en mer plandreven utviklingsmetode har vi, gjennom flittige møter, god kommunikasjon og planlegging, fulgt verdiene og prinsippene i det smidige manifestet.

Kjernefunksjonene som var satt som krav til prosjektet er blitt implementert. Det er fremdeles noen ønskede funksjoner som er delvis utviklet, men som ikke virker helt som ønsket. En av disse funksjonene er opplasting av bilder av spillerne, og mulighet for å oppdatere/endre bilder etter spilleren er lagt til databasen. Dette er en funksjon som viste seg å være vanskeligere enn først antatt, men er allerede delvis implementert i det nåværende systemet.

Søkefunksjonen fungerer på alle sider og det er mulig å søke etter tre forskjellige parametere; navn, beskrivelse og posisjon. Hvis det er ønskelig å videreutvikle denne funksjonen til å kunne søke opp mot flere parametere er det tilrettelagt for, og det skal være enkelt å iverksette. Med vår egen søkefunksjon og metodene vi har implementert gjennom bibliotek for søk og sortering, skal dette være en god løsning på kravet om søkefunksjonalitet.

Som oppdragsgiver har lagt vekt på både for oss og offentlig, er spillerlogistikk et område klubben virkelig ønsker å satse på i årene framover. Det er derfor viktig for oss at produktet kan være til hjelp fra dag en, men også at det skal være et godt grunnlag for videreutvikling. Produktet er funksjonelt og automatiserer mange av oppgavene som oppdragsgiver før har brukt mye tid på å utføre manuelt.

Ønsker arbeidsgiver å videreutvikle produktet, mener vi at dette er en god grunnmur for et styringssystem som kan håndtere flere oppgaver; som å lage nye formasjonsvisualiseringer, håndtere spillerlønn, opprette spillerkontrakter og mer.

7 References

Anon., u.d. *AJAX(programmering); Wikipedia.* [Internett]
Available at: https://no.wikipedia.org/wiki/Ajax_(programmering)
[Funnet 10 Mai 2019].
Anon., u.d. *Bitbucket; Wikipedia.* [Internett]
Available at: https://en.wikipedia.org/wiki/Bitbucket
[Funnet 13 Mai 2019].
Anon., u.d. *Flask.* [Internett]
Available at: http://flask.pocoo.org/
[Funnet 10 Mai 2019].
Anon., u.d. *Foreword; Flask.* [Internett]
Available at: http://flask.pocoo.org/docs/1.0/foreword/
[Funnet 10 Mai 2019].
Anon., u.d. *General Information; dev.mysql.com.* [Internett]
Available at: https://dev.mysql.com/doc/refman/8.0/en/introduction.html
[Funnet 10 Mai 2019].
Anon., u.d. *Github.* [Internett]
Available at: https://github.com/
Anon., u.d. *HTML & CSS; w3.org.* [Internett]
Available at: https://www.w3.org/standards/webdesign/htmlcss
[Funnet 10 Mai 2019].
Anon., u.d. *HTML; Wikipedia.* [Internett]
Available at: https://en.wikipedia.org/wiki/HTML
[Funnet 10 Mai 2019].
Anon., u.d. *JavaScript; Wikipedia.* [Internett]
Available at: https://en.wikipedia.org/wiki/Javascript
[Funnet 10 Mai 2019].
Anon., u.d. *jQuery introduction; w3schools.* [Internett]
Available at: https://www.w3schools.com/jquery/jquery_intro.asp
[Funnet 13 Mai 2019].
Anon., u.d. *Mercurial.* [Internett]
Available at: https://www.mercurial-scm.org/
Anon., u.d. *MySQL Workbench; MySQL.* [Internett]
Available at: https://www.mysql.com/products/workbench/
Anon., u.d. *Product; Bitbucket.* [Internett]
Available at: https://bitbucket.org/product/
Anon., u.d. *Pycharm; JetBrains.* [Internett]
Available at: https://www.jetbrains.com/pycharm/
Anon., u.d. *Python Introduction; w3schools.* [Internett]
Available at: https://www.w3schools.com/python/python_intro.asp
[Funnet 10 Mai 2019].
Anon., u.d. *Python; Wikipedia.* [Internett]
Available at: https://no.wikipedia.org/wiki/Python
[Funnet 10 Mai 2019].
Anon., u.d. *Sourcetree.* [Internett]
Available at: https://www.sourcetreeapp.com/
Anon., u.d. *techopedia.* [Internett]
Available at: https://www.techopedia.com/definition/22801/variable-character-field-varchar
[Funnet 19 Mai 2019].
Anon., u.d. *w3schools.* [Internett]
Available at: https://www.w3schools.com/tags/ref_httpmethods.asp
[Funnet 20 Mai 2019].

Anon., n.d. *What is CSS?.* [Online]
Available at: https://www.w3schools.com/whatis/whatis_css.asp
[Accessed 17 mai 2019].
Anon., n.d. *What is HTML?.* [Online]
Available at: https://www.w3schools.com/whatis/whatis_html.asp
[Accessed 17 mai 2019].
Anon., u.d. *What is MySQL; dev.mysql.com.* [Internett]
Available at: https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html
[Funnet 10 Mai 2019].
Anon., u.d. *What is Unified Modeling Language (UML)?.* [Internett]
Available at: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-uml/
[Funnet 17 mai 2019].
Anon., u.d. *Wikipedia.* [Internett]
Available at: https://en.wikipedia.org/wiki/Salt_(cryptography)
[Funnet 10 Mai 2019].
Anon., u.d. *Wikiversity.* [Internett]
Available at: https://en.wikiversity.org/wiki/Plan-driven_software_development
[Funnet 10 Mai 2019].
Armin Ronacher, u.d. *Flask.* [Internett]
Available at: http://flask.pocoo.org/
[Funnet 10 Mai 2019].
Bayer, M., u.d. *SQLAlchemy; Python Package Index.* [Internett]
Available at: https://pypi.org/project/SQLAlchemy/
[Funnet 12 Mai 2019].
Beck, K. et al., u.d. *Manifestet for smidig programvareutvikling.* [Internett]
Available at: http://agilemanifesto.org/iso/no/manifesto.html
[Funnet 9 Mai 2019].
Bolton, D., 2019. *Definition of Int in C, C++ and C#.* [Internett]
Available at: https://www.thoughtco.com/definition-of-int-958297
[Funnet 18 Mai 2019].
Bratbergsengen, k., 2017. *Store Norske Leksikon.* [Internett]
Available at: https://www.codementor.io/garethdwyer/flask-vs-django-why-flask-might-be-better-4xs7mdf8v
[Funnet 19 Mai 2019].
Bratbergsengen, K., 2018. *Store Norske Leksikon.* [Internett]
Available at: https://snl.no/entitet_-_databaser
[Funnet 19 Mai 2019].
Bratbergsengen, K., u.d. *Database; Store Norske Leksikon.* [Internett]
Available at: https://snl.no/database
[Funnet 10 Mai 2019].
Christensson, P., 2006. *String Definition.* [Internett]
Available at: https://techterms.com/definition/string
[Funnet 18 Mai 2019].
CodesCracker, u.d. *HTML Document Structure.* [Internett]
Available at: https://codescracker.com/html/html-structure.htm
[Funnet 19 Mai 2019].
contributors, W., 2019. *Bitbucket.* [Internett]
Available at: https://en.wikipedia.org/wiki/Bitbucket
[Funnet 13 Mai 2019].
Crockford, D., u.d. *Json.* [Internett]
Available at: https://www.json.org
[Funnet 19 Mai 2019].
Gkogka, E., u.d. *Gestalt principles in UI design; Muzli Magazine.* [Internett]
Available at: https://medium.muz.li/gestalt-principles-in-ui-design-6b75a41e9965
[Funnet 9 Mai 2019].

Gordon, A. & Hernandez, S., 2016. *The Official (ISC)2 Guide to the SSCP CBK.* s.l.:John Wiley & Sons.
Hope, C., 11. *computerhope.* [Internett]
Available at: https://www.computerhope.com/jargon/t/tag.htm
[Funnet 19 mai 2019].
Johansson, T. & Crasta, J., u.d. *WTForms; Python Package Index.* [Internett]
Available at: https://pypi.org/project/WTForms/
[Funnet 12 Mai 2019].
Mudrinic, S., 2018. REIDAR VÅGNES TIL AAFK. *Aalesunds FK,* 23 50.Volum 2018.
MySQL™, 2019. *General Information.* [Internett]
Available at: https://dev.mysql.com/doc/refman/8.0/en/introduction.html
[Funnet 10 Mai 2019].
MySQL™, 2019. *What is MySQL.* [Internett]
Available at: https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html
[Funnet 10 Mai 2019].
MySQL™, u.d. *MySQL Workbench.* [Internett]
Available at: https://www.mysql.com/products/workbench/
[Funnet 16 Mai 2019].
Pfahl, D., 2014. *Software Engineering,* s.l.: s.n.
Ronacher, A., u.d. *Werkzeug; Python Package Index.* [Internett]
Available at: https://pypi.org/project/Werkzeug/
[Funnet 12 Mai 2019].
Rouse, M., 2005. *Search sql server.* [Internett]
Available at: https://searchsqlserver.techtarget.com/definition/primary-key
[Funnet 19 Mai 2019].
Rouse, M., 2017. *Tech Target.* [Internett]
Available at: https://searchoracle.techtarget.com/definition/foreign-key
[Funnet 19 Mai 2019].
Shneiderman, B., 1986. *Designing the User Interface: Strategies for Effective Human-Computer Interaction.* s.l.:Pearson.
Shneiderman, B., 2016. *The Eight Golden Rules of Interface Design.* [Internett]
Available at: http://www.cs.umd.edu/~ben/goldenrules.html
[Funnet 9 Mai 2019].
Sommerville, I., 2016. *Software Engineering.* s.l.:Pearson.
Sommerville, I., 2016. *Software Engineering.* Global Edition; Tenth Edition red. s.l.:Pearson Higher Ed.
SpryMedia Ltd., u.d. *DataTables.* [Internett]
Available at: https://datatables.net/
[Funnet 18 Mai 2019].
Stenerud, K., 2018. AaFK har tatt store grep rundt spillerlogistikken. *Sunnmørsposten,* 06 September.Volum 2018.
The Interaction Design Foundation, u.d. *Gestalt Principles; The Interaction Design Foundation.* [Internett]
Available at: https://www.interaction-design.org/literature/topics/gestalt-principles
[Funnet 9 Mai 2019].
University of Maryland, u.d. *University of Maryland.* [Internett]
Available at: http://www.cs.umd.edu/~ben/
[Funnet 9 Mai 2019].
W3C, 2016. *HTML & CSS.* [Internett]
Available at: https://www.w3.org/standards/webdesign/htmlcss
[Funnet 10 Mai 2019].
w3schools, u.d. *jQuery introduction.* [Internett]
Available at: https://www.w3schools.com/jquery/jquery_intro.asp
[Funnet 13 Mai 2019].
w3schools, u.d. *SQL Injection.* [Internett]
Available at: https://www.w3schools.com/sql/sql_injection.asp
[Funnet 10 Mai 2019].

w3schools, u.d. *SQL Injection; w3schools.com.* [Internett]
Available at: https://www.w3schools.com/sql/sql_injection.asp
[Funnet 10 Mai 2019].
w3shools, u.d. *Python Introduction.* [Internett]
Available at: https://www.w3schools.com/python/python_intro.asp
[Funnet 10 Mai 2019].
Whitman, M. E. & Mattord, H. J., 2017. *Principles of Information Security.* s.l.:Cengage Learning.
Whitman, M. E. & Mattord, H. J., 2017. *Principles of Information Security.* Sixth Edition red. s.l.:Cengage Learning.
Wikipedia contributors, 2019. *HTML.* [Internett]
Available at: https://en.wikipedia.org/wiki/HTML
[Funnet 10 Mai 2019].
Wikipedia contributors, 2019. *JavaScript.* [Internett]
Available at: https://en.wikipedia.org/wiki/Javascript
[Funnet 10 Mai 2019].
Wikipedia contributors, 2019. *Social engineering (security).* [Internett]
Available at: https://en.wikipedia.org/wiki/Social_engineering_(security)
[Funnet 10 Mai 2019].
Wikipedia-brukere, 2016. *AJAX(programmering).* [Internett]
Available at: https://no.wikipedia.org/wiki/Ajax_(programmering)
[Funnet 10 Mai 2019].
Wikipedia-brukere, 2017. *Structured Query Language.* [Internett]
Available at: https://no.wikipedia.org/wiki/Structured_Query_Language
[Funnet 17 mai 2019].
Wikipedia-brukere, 2019. *Python.* [Internett]
Available at: https://no.wikipedia.org/wiki/Python
[Funnet 10 Mai 2019].
Wikiversity contributors, 2019. *Plan-driven software development.* [Internett]
Available at: https://en.wikiversity.org/wiki/Plan-driven_software_development
[Funnet 10 Mai 2019].

Vedlegg

Vedlegg 1 Forprosjektrapport
Vedlegg 2 Møtereferat
Vedlegg 3 Prototyper

Vedlegg 1

Forprosjektrapport

TITTEL:

Spillerdata-behandling

KANDIDATNUMMER(E):

Espen Bårdevik, Fredrik Kvalheim, Storm Østberg

DATO:	EMNEKODE: *	EMNE:	DOKUMENT TILGANG:
01.02.19	**IE303612**	**Bacheloroppgave (Data)**	- Åpen

STUDIUM:	ANT SIDER/VEDLEGG:	BIBL. NR:
INGENIØRFAGET DATA	9 / 0	- Ikke i bruk -

OPPDRAGSGIVER(E)/VEILEDER(E):

Aalesund fotball klubb / Di Wu, Anniken Karlsen

OPPGAVE/SAMMENDRAG:

Denne oppgaven er en eksamensbesvarelse utført av studenter ved NTNU i Ålesund i samarbeid med AaFK's sportslige leder Bjørn Erik Melland og våre veiledere Di Wu og Anniken Karlsen. Oppgaven er å lage et oversiktlig system for å opprette og lagre spillerlogistikk for AaFK, og som kan brukes i flere år fremover.
Systemet er ment til å gi AaFK en fordel fremfor sine konkurrenter.

INNHOLD

1 INNLEDNING

Vi er en gruppe på tre dataingeniør studenter.

Vi valgte denne oppgaven fordi den virker overkommelig uten å være for liten, og kan utvides basert på hvor detaljert data som er ønskelig. Visualiseringsdelen av oppgaven virker også særdeles spennende.

I gruppen er vi også flere som er fotballinteresserte og vi liker alle å spille fotball når vi kan.

Vår oppgave er gitt av AaFK's sportslige leder Bjørn Erik Melland. Han vil ha en måte for AaFK å samle all spillerdata på samme plass. Slik som Melland jobber nå ligger mye av informasjonen splittet på forskjellige plasser på datamaskinen og krever mye unødvendig arbeid for å sammenligne og å hente frem forskjellig informasjon.

Formålet med oppgaven er å opprette en database som skal kunne håndtere all den dataen som AaFK trenger, og å gi dem et enkelt brukergrensesnitt til å hente og vise det frem på en enkel og fancy måte.

2 BEGREPER

Jira – et verktøy for delegering av oppgaver som gir en rapport på hvordan arbeidet er blitt utført.

Git – Versjonskontrollsystem for lagring av filer

NetBeans – Programvareutviklings-plattform, brukt for å skrive kode

PyCharm – Programvareutviklings-plattform, brukt for å skrive kode i Python

Python – Objektorientert programmeringsspråk

PuTTY – Terminal emulator, seriell-konsoll og nettverk filoverføring program

Zotero – Programvare for lagring av referanser

Agile – Fleksibel arbeidsmetode hvor du planlegger arbeidsoppgaver i sprints

Sprints – Korte arbeidsperioder med bestemte arbeidsoppgaver som skal bli utført innen fastsatt tid

Skygge-lag (Shadowteam) – Representasjon av spillere i designerte posisjoner på banen.

3 PROSJEKTORGANISASJON

3.1 Prosjektgruppe

Studentnummer(e)
274945
476522
476519

3.1.1 Oppgaver for prosjektgruppen - organisering

Gruppen må organisere seg slik at alle kravene blir møtt. For å garantere dette skal vi bruke en agile arbeidsplan og velge arbeidsoppgaver etter hvor stor arbeidsmengde den enkelte klarer å gjøre i løpet av en sprint

3.1.2 Oppgaver for prosjektleder

Prosjektleder skal forsikre seg at alle gjør sin del av oppgaven og holde seg oppdatert på hvordan oppgaven går i forhold til målene.

Han skal sette opp nye oppgaver/problemer og gi arbeidsoppgaven en poengsum etter vanskelighetsgrad.

3.1.3 Oppgaver for sekretær

Fortløpende rapportskriving, logistikk av møter og potensielle dedikerte sekretæroppgaver.

Dette blir hovedsakelig gjort som en gruppe. De aller fleste møtene er allerede satt opp, alle møter med veileder er satt fast annenhver uke og det samme med kontaktperson i AaFK.

3.1.4 Oppgaver for øvrige medlem(mer)

Alle medlemmer har ansvar om å fullføre sine tildelte oppgaver innen gitt tid, eventuelt diskutere og finne løsning med prosjektleder om hvordan det skal håndteres hvis oppgaven(e) viser seg å ta lenger tid en originalt antatt.

Alle skal møte til avtalt tid til møter, evt. informere om det ikke er mulig.

3.2 Styringsgruppe (veileder og kontaktperson oppdragsgiver)

Veileder: Di Wu

Biveileder: Anniken Karlsen

Kontaktperson AaFK: Bjørn Erik Melland

4 AVTALER

4.1 Avtale med oppdragsgiver

Etter møte med Bjørn Erik Melland, kontaktperson, fikk vi vite at oppgaven var litt annerledes vi først antok fra oppgaveteksten vi fikk utdelt.

Vi diskuterte ønsket kjernefunksjonalitet, brukere og roller.

Ønsket funksjonalitet:

- Lage et system for håndtering av spillerlogistikk med vektlegging på spillerrekrutering og speiding.
- En måte å lagre spillerprofiler på ett strukturert vis.
- Mulighet for å lage og endre spillerprofiler fra administrator og bruker-siden.
- Lage ett “skygge-lag”, som presenterer mulige spillere i designerte roller.
- Lage en metode for å rangere speidete spillere etter interesse.
- Lage en søkefunksjon/metode for sortering som gjør det enkelt å finne ønskede spillere.

Brukere og roller:

- Sportslig leder – Administrator. Full tilgang til å se, opprette og endre spillerdata.
- Speider – Scout. Tilgang til å se, opprette og endre spillerdata til egne spillere.

Videre møter er satt til fredager kl. 12.00 annenhver uke på Color Line Stadion. Sprintene i vår agile arbeidsplan vil bli planlagt deretter.

4.2 Arbeidssted og ressurser

Arbeidssted(er): Lab rom L167 tirsdag til fredag og NMK.

Tilgjengelige ressurser: skolens arbeidsplasser, veiledere, arbeidsgiver, programlisenser fra NTNU.

Avtalt rapportering: møter med veiledere annenhver tirsdag og bedrift annenhver fredag.

4.3 Gruppenormer – samarbeidsregler – holdninger

Alle medlemmer i gruppen skal jobbe med prosjektet, og gjøre de oppgavene de har blitt gitt i Jira.

Det er forventet at alle skal jobbe jevnlig med prosjektet og legge inn den krevde arbeidsmengden for å fullføre arbeidsoppgavene til avtalte tider.

Koden skal skrives modulært, det vil si at kodesnutter skal være selvstendige, og ikke avhenge av andre blokker av kode. Dette gjør systemet lettere å vedlikeholde og å utbygge i senere tid.

Kommentering av koden skal gjøre det lettere å forstå vanskelige funksjoner uten å bli for detaljert.

Dette er alle viktige prinsipper gjennom utviklingen av programvare.

5 PROSJEKTBESKRIVELSE

5.1 Problemstilling - målsetting - hensikt

Per dags dato oppretter Bjørn Erik Melland mange forskjellige dokumenter for å håndtere spillerlogistikk. Disse dokumentene blir organisert i forskjellige mapper som han må navigere gjennom hver gang han skal enten oppdatere informasjonen eller skal vise den frem. Dette involvere ofte mye navigering frem og tilbake, og er et system Melland kan, men som neste man vil se på som kaos, og muligens må lage seg et eget system igjen. Og alt er lagret på datamaskinen hans.

Melland vil ha et system som forenkler hele prosessen med både å lage og oppdatere eksisterende spillerprofiler. Han vil at systemet skal være intuitivt slik at hvis noen skal ta over rollen til Melland skal de føle at det er et system de kan fortsette på og slippe å lage sitt eget.

Vi vil forenkle dette systemet som Melland bruker og minimere arbeidsmengden for å vedlikeholde det. Dette vil vi gjøre med å lage en nettside hvor all denne informasjonen kan plottes inn og vises frem på en oversiktlig og fancy måte.

Informasjonen skal lagres på en server slik at det ikke er låst fast til en enkelt datamaskin, men til brukere.

Målet er å lage et system som AaFK kan fortsette å bruke i mange år fremover og som kan gå i «arv», dvs. personen som tar over rollen til Melland kan gå rett inn i og fortsette å bruke systemet, uten at de må organisere all spillerlogistikk på nytt.

5.2 Krav til løsning eller prosjektresultat – spesifikasjon

Vår løsning må inneholde en database med struktur som tillater lagring av all spillerdata bedriften trenger, for eksempel informasjon om: rolle, alder, speider, agent, kontraktlengde og navn.

Vi trenger også en nettbasert applikasjon/nettside som kan brukes til rapportering og representering av spillerdata.

Videre funksjonelle krav:

- Lage en søkefunksjon/metode med sortering som gjør det enkelt å finne ønskede spillere.
- Lage en metode for å rangere speidet spillere etter interesse.
- Lage ett "skygge-lag", som presenterer mulige spillere i designerte roller.

5.3 Planlagt framgangsmåte(r) for utviklingsarbeidet – metode(r)

Vi bruker JIRA som en agile arbeidsplan, hvor vi setter opp arbeidsoppgaver i to-ukers planer, kalt sprints.

Ved å bruke en slik arbeidsplan vil vi få en strukturert arbeidsplan som vi kan tilpasse etter behov. Vi kan planlegge slik at vi skal være ferdige med bestemte økter slik at vi kan vise til fremgang ved hvert møte med bedriften.

5.4 Informasjonsinnsamling – utført og planlagt

Veiledere og kontaktperson hos AaFK for informasjon rundt oppgaven, og tips til prosjekt planlegging.

Bostrøm, *Edgar (2000), Datamodellering: praksis og teori.* Centraltrykkeriet AS, Sarpsborg.

Planlagt informasjonsinnsamling: Kontaktperson i AaFK og veileder for hjelp, veiledning og informasjon angående oppgavens utforming.

5.5 Vurdering – analyse av risiko

De største risikoene vi står ovenfor med dette prosjektet gjelder sikkerhet og tidsbruk.

Prosjektet vil ha en basisfunksjonalitet som, med mindre vi møter uventede komplikasjoner, skal være overkommelig å komme i mål med i god tid.

Vår største utfordring når det gjelder tid, ligger i å sørge for at sikkerheten til systemet er tilstrekkelig.

Sikkerhet er svært viktig for dette systemet, da det skal holde på sensitiv informasjon. Sikkerheten må være på plass før systemet kan bli tatt i bruk. Dette er noe som kan vise seg å kreve mye arbeid og kan gå ut over tid vi trenger får å implementere all ønsket funksjonalitet.

Ekstra funksjonalitet, f.eks. "skygge-laget", som er en visuell representering av spillere på banen i sine posisjoner, kan sees på som en bonusfunksjonalitet og vil ikke være førsteprioritet. Slik funksjonalitet vil derfor stå i fare hvis basisfunksjonaliteten eller implementasjon og testing av sikkerhet tar lengre tid enn planlagt.

5.6 Hovedaktiviteter i videre arbeid

Enkeltoppgaver vil bli utdelt underveis, ettersom vi setter opp to-ukers planene. Derfor vil ansvar være ikke tildelt (IT) på enkeltoppgaver nå.

FK – Fredrik Kvalheim
SØ - Storm Østberg
EB – Espen Bårdevik

NR.	Navn	Ansvar	Kostnad	Ca. tid
A1	Lage server	FK	Usikkert	2 uker
1.1	Opprette server	FK	Usikkert	1 uke
1.2	Implementere	FK	-	1 uke

	serversikkerhet			
A2	Lage database	IT	-	2 uker
2.1	Lag tabeller	IT	-	1 uke
2.2	Lag relasjoner	IT	-	1 uke
A3	Lage nettside/web applikasjon	IT	-	5 uker
3.1	Lage et grensesnitt	IT	-	2 uker
3.2	Legge til funksjonalitet	IT	-	2 uker
3.3	Lage side for spillerprofiler	IT	-	1 uke
3.4	Lage side for speiderprofiler	IT	-	2 dager
3.5	Lage side for agentprofil	IT	-	2 dager
3.6	Lage søkefunksjon	IT	-	1 uke
A4	Implementer sikkerhet	IT	-	2 uker
4.1	Hindre vanlige angrep	IT	-	1 uke
4.2	Sikre passordet (kryptere og salte)	IT	-	1 uke
A5	Legge til mulighet for å legge til/endre spillerdata	IT	-	2 uker
5.1	Sende informasjonen til serveren	IT	-	3 dager
5.2	Bekrefte eller avkrefte om informasjonen ble sendt inn riktig	IT	-	3 dager
5.3	Lage en god visuell representasjon	IT	-	1 uke
A6	Lage “skygge-lag”	IT	-	3 uker
6.1	Lage en egen side for skygge-laget	IT	-	2-3 dager
6.2	Lage en måte å presentere spillerne visuelt i et lag	IT	-	2 uker
6.3	La en bruker legge til eksisterende spillere til posisjoner	IT	-	3-4uke

A7	Design og små detaljer	IT	-	Resterende tid
Total			Usikkert	16 uker

Tabellen viser planlagte oppgaver for prosjektet, vi er usikre på hvilke kostnader serveren vil påføre, om noen, så det er listet som usikkert.

5.7 Framdriftsplan – styring av prosjektet

5.7.1 Hovedplan

Bruke Jira for å jobbe så effektivt som mulig ved å velge deloppgaver etter evne og interesse, her er det viktig å lage oppgaver som er mulige å gjennomføre i løpet av en sprint og å dele opp store oppgaver i flere mindre oppgaver som e mulige å fullføre.

Underaktivitetene beskrevet i 5.6 fordeles fortløpende etter behov gjennom arbeidsfordelingsmetoden Jira.

Annenhver uke skal vi ha møte med arbeidsgiver, arbeidsplanen er derfor planlagt slik at vi skal ha fullført en sprint til hvert møte, slik at vi alltid har noe nytt å vise til arbeidsgiver.

Vi må gjøre beslutninger underveis, i tilfelle enkelte oppgaver tar lengre tid enn vi har forventet, angående ekstra funksjonalitet som eventuelt må nedprioriteres eller bli kuttet ut fullstendig.

5.7.2 Styringshjelpemidler

For å styre prosjektet skal vi bruke Jira slik at den enkelte kan hente ut oppgaver og de andre vil bli informert om denne personenes fremgang på den oppgaven. Dette vil gjøre det lettere å sørge for at ikke flere personer jobber med samme deloppgave og vil forhindre forvirring og uenigheter.

Vi lager UML-diagram for å ha en oversikt over hvordan databasen vil bli seende ut.

Vi tenker også å benytte møter med veiledere som en del av den agile arbeidsplanen ved å gå over arbeidet man har gjort den følgende sprinten og se hvor effektivt man klarer å jobbe.

5.7.3 Utviklingshjelpemidler

Jira
PyCharm
NetBeans
Zotero
Git
PuTTY
NTNU SharePoint
Draw.io

5.7.4 Intern kontroll – evaluering

Gjennom Git (versjonskontroll programvare) kan vi se over endringer i koden før den blir publisert til en versjon vi viser frem, dette gjør at vi alltid vil ha en fungerende versjon å gå tilbake til om det skulle forekomme feil. Den fungerende versjonen vil være hva vi viser frem i møte med veiledere og kontaktperson i bedrift.

Kriteriene for at endringer blir endelig er at de fungerer som de skal og ikke endrer funksjonaliteten til andre tidligere endringer.

5.8 Beslutninger – beslutningsprosess

Beslutninger om store endringer i arbeidsprosessen vil bli tatt internt først, for så å kontakte veiledere om hvorvidt dette kan være en mulighet, og for å få eventuelle innspill.

Beslutninger til oppgaven (ikke tekniske løsninger) tas med bedrift slik at bedriften kan bli fornøyd med sluttproduktet.

6 DOKUMENTASJON

6.1 Rapporter og tekniske dokumenter

Dokumentasjonen vil bli gjort fortløpende for å beholde presisjonen i detaljer angående arbeidsprosess, tekniske detaljer og problemer.

Vi har enda ikke utarbeidet noen rutiner angående rapportering, men den enkelte er ansvarlig for å logge det de gjør slik at ingen kritisk informasjon blir utelatt. Bruker filosofien med at det er bedre med for mye enn for lite.

For å distribuere applikasjonen vil vi gi en link til AaFK og sørge for at de har tilgang til alle systemer, AaFK må selv stå for kostnaden av leie på server. Slik vil AaFK selv ha tilgang til det baken liggende om endringer senere skal bli gjort.

Vi ønsker i denne tid ikke at alle skal ha tilgang til nettsiden slik at den er mindre utsatt for eventuelle angrep.

Systemet skal utvikles slikt at det skal kreves minst mulig vedlikehold, men sørge for at bedriften kan få en fordel over konkurrenter. Dette betyr at det i fremtiden kan være behov for å legge til funksjonalitet eller utvide/forenkle programmet.

7 PLANLAGTE MØTER OG RAPPORTER

7.1 Møter

7.1.1 Møter med styringsgruppen

Planlagte møter er annenhver uke på tirsdager, med veileder, og fredager med bedrift.

Dette gjøres for å kunne få oppklaring på spørsmål, oppfølging på arbeidsprosessen og kritikk på feil i produktet.

7.1.2 Prosjektmøter

Prosjektmøter vil bli samkjørt med styremøter.

7.2 Periodiske rapporter

Vi vil kontinuerlig arbeide med å rapportere på det vi arbeider med slik at vi kan garantere at informasjonen i sluttrapporten blir mest mulig nøyaktig.

7.2.1 Framdriftsrapporter (inkl. milepæl)

Framdriftsrapporter blir gjennomført på møter med veiledere og kontaktperson i AaFK, siden dette er en del av den agile arbeidsmetoden vi bruker.

8 PLANLAGT AVVIKSBEHANDLING

Ved avvik fra fremdriftsplanen vil vi fokusere på kritiske funksjoner slik at sluttproduktet skal være mest mulig funksjonelt for AaFK.

For større endringer vil vi ta kontakt med veiledere for å få tips for mulige løsninger på eventuelle problemer og en ny eventuell fremdriftsplan

9 UTSTYRSBEHOV/FORUTSETNINGER FOR GJENNOMFØRING

PyCharm, Jira og NTNU SharePoint dette er lisenser NTNU holder som vi benytter for tilgang til programvarene.

Vedlegg 2

Møtereferater

Planlegging til møte 01.02.19

Spørsmål til bedrift

Hvem er kontaktpersonen vår (kontaktinformasjon)?

Er det greit om vi tar opp samtalen, slik at vi kan gå gjennom å bekrefte detaljer senere?

Hva er det eksisterende nettverket bestående av?

Mulig å se strukturen til database/hvordan skal databasen hostes?

Muligheter for å jobbe med prosjektet on-site?

Databaseplattform (payara, glassfish, java EE)?

Noe vi må ta forbehold til?

Hvem skal bruke systemet?

Krav

- Hva MÅ være med?
- Hva er evt. ønsket?
- Hvem skal ha tilgang (rolletillatelser)?
- Noen begrensinger? (hva er det eksisterende)
- Bruke eksisterende rammeverk eller utvikle nye?
- Noen bestemte søke begrep?
- Hva skal vi hente ut informasjon, og hvilken informasjon er ønsket fra datasettet?
- Skal databasen integreres med noen andre teknologier?
- Hvilke begrensinger har de forskjellige brukerene? (brukergruppene)

Brukergrensesnitt

- Enheter som skal bruke grensesnittet?
- Ønsket funksjonalitet?
- Grafisk utforming (farger, stil)

Møtereferat 01.02.19

Personer til stede: Espen Nordfjellmark Bårdevik, Fredrik Halsebakke Kvalheim, Storm Østberg og Bjørn Erik Melland

Fastslo at oppgaven var en del annerledes enn det vi trodde.

Diskuterte ønsket kjernefunksjonalitet, brukere og roller.

Ønsket funksjonalitet:

- Lage et system for håndtering av spillerlogistikk med vektlegging på spillerrekrutering og speiding.
- Mulighet for å lagre spillerprofiler på ett strukturert vis.
- Mulighet for å lage og endre spillerprofiler fra administrator og klient-siden.
- Lage ett "skygge-lag", som presenterer mulige spillere i designerte roller.
- Måte å rangere interesse for speidete spillere.
- Lage en søkefunksjon/metode for sortering som gjør det enkelt å finne ønskede spillere.

Brukere og roller:

1. Sportslig leder – Administrator. Full tilgang til å se, opprette og endre spillerdata.
2. Speider – Full tilgang til å se opprette og endre spillerdata til egne spillere.

Krav til oppgaven

Ting som må være med:

- Database til å lagre spillerdata
- Sikker innlogging, kun autorisert innlogging (Speider/Administrator)
- Forskjellige brukere
- Opprette og endre spillerprofil
- Mulighet til å søke i database

Middels prioritet:

- Lage skyggelag funksjon

Lav prioritet:

- Lage en «hjelp» fane for forskjellig informasjon som kanskje ikke er selvsagt.

Andre egenskaper som kan være ønskelige:

- Lage grupper av spillerstallen for å sammenligne kostnad og diverse andre ting (A-lag, B-lag, junior lag)
- Kunne få ut gjennomsnitt av forskjellige ting (lønn, alder, spilletid)

- Koble til informasjon fra transfermarkt til nettsiden, og muligens andre nettsider
- Kunne hente ut forskjellige lister etter spesifikke krav (alder, høyde, lønn, posisjon etc.)
- Skal kunne lage et ferdig prospekt med å trykke på en knapp på en spillerprofil

Møte 01.02.19

Personer til stede: Espen Nordfjellmark Bårdevik, Fredrik Halsebakke Kvalheim og Storm Østberg.

Opprettet et UML-diagram for databasen. Diskuterte entiteter, attributter og lister.

Sendte mail til Bjørn-Erik for oppklaring rundt ønskede attributter.

Lagde konsept for spiller-, agent- og speiderprofil.

Diskuterte back-end språk - foreløpig konklusjon: python.

Møte 05.02.19

Personer til stede: Espen Nordfjellmark Bårdevik, Fredrik Halsebakke Kvalheim, Storm Østberg, Di Wu og Anniken Susanne T. Karlsen

Forbedret og oppdatert UML-diagram. Endret mange-til-mange forhold til en-til-mange med nye lister. La til Club, ScoutedPlayer og PlayerPosition-lister.

La vekt på loggføring og dokumentering av møter og viktige dokumenter.

Møte 15.02.19

Personer til stede: Espen Nordfjellmark Bårdevik, Fredrik Halsebakke Kvalheim, Storm Østberg og Bjørn Erik Melland

Viste frem noen konsept på hvordan vi kunne prøve å få nettsiden til å se ut for en bestemt bruker.

Fikk gitt:

- Eksempel på hvordan et prospekt ser ut per dags dato
- Fikk mer utdypning på hvordan fargekoder blir brukt i Excel
    - Grønn = spennende
    - Gul = under oppfølging
    - Grå = inviter til gruppespill
    - Hvit = ikke gjort noe med
- Fikk utskrift av hvordan laget må se ut i forhold til lokale og utenlandske spillere.

Vi trenger ingen avtale om konfidensialitet så lenge vi bruker oppfunnet informasjon i oppgaven vi leverer inn.

Vi ønsker mulighet til å hente informasjon fra i transfermarkt/soccerway/instatscout for historikk og spilleraktivitet. Eventuelt muligheter for å legge inn lenker til disse sidene slik at speidere kan legge inn de lenkene de føler er relevante.

Bjørn Erik ønsker å kunne se lønn på spillere og kunne redigere og se hvor mye det frigis i kapital av å selge en spiller og kjøpe noen andre. Ønsker også å kunne lage grupper og legge spillere til disse (favoritter, A-lag, Bærebjelker ...) en spiller skal kunne være med i flere grupper.

Bjørn Erik virker veldig fornøyd med fremgangen.

Møte 01.03.2019

Personer til stede: Espen Nordfjellmark Bårdevik, Fredrik Halsebakke Kvalheim, Storm Østberg og Bjørn Erik Melland

Design og fargekoding av grafisk brukergrensesnitt bør nedprioriteres, da et enkelt, men intuitivt design er tilstrekkelig. Oppdragsgiver ønsker å bruke eget team for videreutvikling av design.

Oppdragsgiver ønsker å kunne søke på spillere etter flere parametere.

Spurte om tekniske krav for serveren.

Møte 05.03.2019

Personer til stede: Espen Nordfjellmark Bårdevik, Fredrik Halsebakke Kvalheim, Storm Østberg og Di Wu

Bør spre informasjonen til "Player" til flere mindre tabeller, i stedet for å ha all informasjon i samme tabell.

Bestemte oss for å dele opp informasjon i to nye tabeller. En tabell skal holde på generell informasjon om en spiller (fødselsdato, navn, høyde, osv.), den andre skal holde på informasjon om en spillers karriere (nåværende klubb, lønn, lengde på kontrakt, etc.).

Møte 03.05.2019

Personer til stede: Espen Nordfjellmark Bårdevik, Fredrik Halsebakke Kvalheim, Storm Østberg og Bjørn Erik Melland

Nye ønsker til prosjektet:

Legge til ny beskrivelse til eksisterende spiller, mulighet til å redigere basisinformasjon.

Mulighet til å fjerne brukere.

Mulighet til å endre passord på brukeren med prompt for å endre passord på førstegangsinnlogging.

Lagre prospekt i databasen med bilder fra transfermark mulighet til å legge inn bilder i prospekt og spillere.

Lage funksjoner for redigering av spiller- og ansattprofiler.

En trener-bruker med mulighet til å se men ikke redigere kan være aktuelt, men har liten hensikt og bør ikke være en prioritering.

Lister under favoritter for å kategorisere, prioritering av spillere i favoritter. Bilde, navn, fødselsdato

Mulighet til å endre posisjoner. (disse kan for øyeblikket sløyfes)

- Ving back
- Sentral midtbane
- Spiss

Mulighet til å hente ut kontaktinfo til speidere og agenter.

Agentliste for å kunne se alle agenter (tlf.nr, mail og navn), samme med speidere.

Mulighet til å slette spillere.

Mulighet til å slette brukere.

Sende mail ved innlegging av spillere.

Sorteringsliste på søk.

Møte 07.05.2019

Personer til stede: Espen Nordfjellmark Bårdevik, Fredrik Halsebakke Kvalheim og Storm Østberg.

Planer videre 07.05.2019:

-Bruke brukersesjoner til å bestemme innhold av sider.

-Endre link til profil - bruk ID ikke navn.

-Favoritter skal komme opp på favoritt-siden.

-Søk skal fungere fra alle sider. Nå fungerer den ikke på spillerlistesiden.

-Endre tittel på nettside til AALESUNDS FK

-Endre styling på profil

-Bruke regex til å forhindre SQL-injeksjoner.

-Endre stylingen til tabeller.

-Fiks lenker på profil så en spiller kan holde på flere lenker.

-Mulighet til for å redigere spillerinfo fra profil.

-Legge inn mulighet for å skrive ut spillerprofiler som prospekter.

-Lage agent- og speiderprofiler som holder på kontaktinformasjon og kan endres fra applikasjonen.

-Endre passord.

Lav prioritet:

-Stallkomposisjon skal vise nåværende spillere som favoritter men delt inn I posisjoner.

-Mulighet til å laste opp bilder, som fil tilknyttet en spiller.

-Mulighet til å laste opp forskjellige filtyper.

-Mulighet til å slette brukere og spillere.

-Legge inn en trenerbruker

-Oppdatere Bjørn ved oppdatering av spillere i form av e-post.

-Fredrik jobber med rapport.

Møte 09.05.2019

Personer til stede: Espen Nordfjellmark Bårdevik, Fredrik Halsebakke Kvalheim og Storm Østberg.

Bør favoritt ha en egen tabell? Slik at en spiller kan være favoritt for noen brukere, men ikke favoritt for andre. Ta opp med oppdragsgiver på neste møte.

Gjenstående oppgaver for utvikling:

-Sørg for at spillerobjekter på favorittsiden linker til spillerprofilen til spilleren.

-Endre på "legg til spiller"-siden så man kan legge til andre klubber enn de som nå er i systemet.

Møte 14.05.2019

Personer til stede: Espen Nordfjellmark Bårdevik, Fredrik Halsebakke Kvalheim, Storm Østberg, Di Wu og Anniken Susanne T. Karlsen

Retningslinjer for videre rapportskriving:

- Oversette begreper om mulig, kan heller bruke engelske begreper i parentes bak.
- Vis om enkelte deler er oversatt, og notere sidetall på referanser.
- Finn bok med html layout.
- Endre bakgrunnen på bilder eller lime inn koden.
- Forklar koden og funksjonaliteten den gir.
- Prosessen er viktig!
- Hør med Di Wu om kildeføring.
- Beskrivelse av systemets arkitektur.
- Bilde av sammenkoblingen av server, database og web-klient.
- Start med å vise spillere på bane først.

Møte 16.05.2019

Personer til stede: Espen Nordfjellmark Bårdevik, Fredrik Halsebakke Kvalheim, Storm Østberg og Bjørn Erik Melland

Få til mest mulig av følgende funksjonalitet (resten blir ansvar for et eventuelt videreutviklingsteam):

Bør kunne legge til flere tidligere klubber, mulighet for det?

Mulig endre stall formasjon for å vise klubbens spillere i formasjon.

Varsle på mail ved ny informasjon.

Mulighet for å skrive ut, viktig for oversikt og kontroll.

Mulighet for å legge inn PDF/Word filer

Send mail med spesifikke spørsmål rundt server, mulighet for tilkobling uten VPN?

Skjerf til presentasjon.

Nødvendig informasjon på spillerlistesiden:

- Navn, fødselsdato, nåværende klubb, agent, posisjon, stikkordsbeskrivelse

Burde vurdere nøkkelordsøk

- Utskriftsfunksjon for liste av spillere (om mulig)

Kunne begrense søk til en dato.

Lagre favoritt-side.

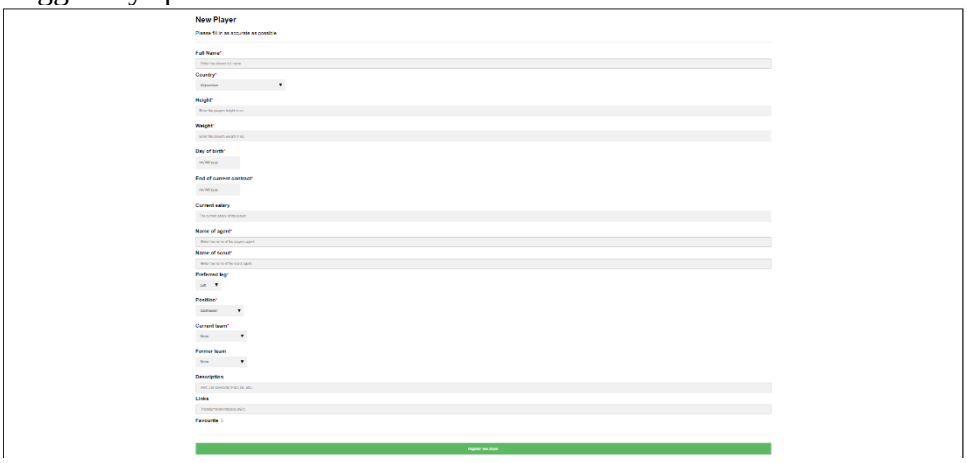
Sortere liste til nyeste først.

Fargekoding for å vise grad av interesse.

Vedlegg 3

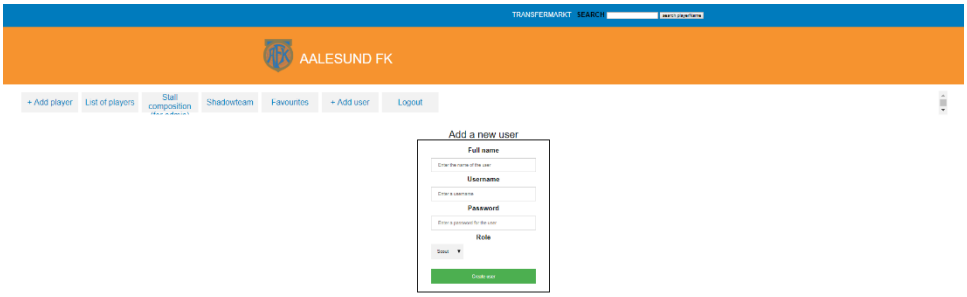
Prototyper

Prototype 1:
Legg til ny spiller.



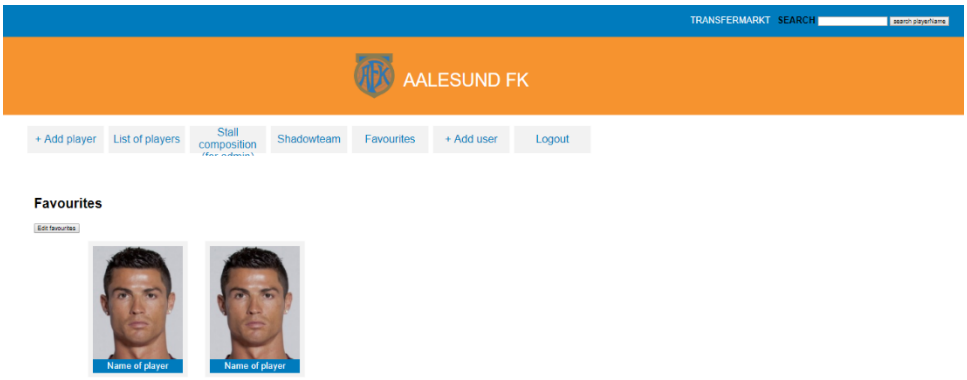
Første funksjonelle utgave av “legg til spiller”-siden. Siden er oversiktlig og fungerer, men denne utgaven hadde kun mulighet til å legge inn en lenke og feltene *posisjon, tidligere klubb og forrige klubb* hadde ingen mulighet til å ta inn nye verdier. Fargen på knappen var grønn for å være tydelig.

Prototype 2:
Legg til ny bruker.



“Legg til bruker”-siden med den gamle grønne knappen.

Prototype 3:
Favorittsiden.



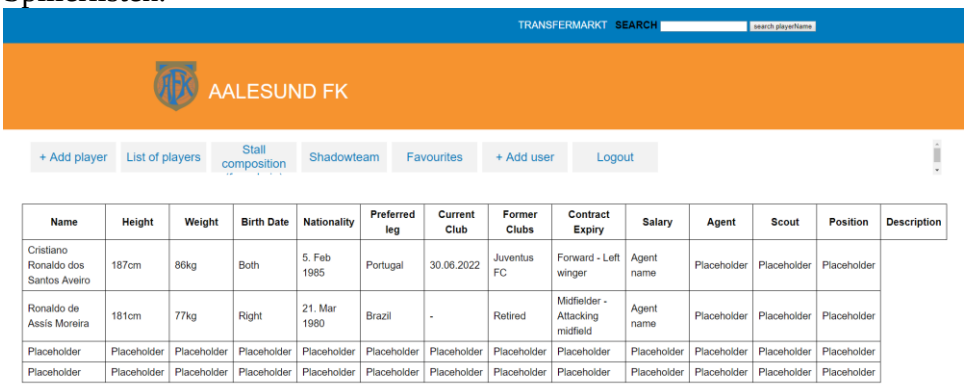
Den første utgaven av favorittsiden viste alle spillerobjekter med bilder som lenket til spillerprofiler. Siden var funksjonell og viste informasjon godt.

Prototype 4:
Kategorisiden.

Category	Demography	Ages	Training cycle	Matches
Mainstays 6+1	4 LUS, herav minimum 1 KUS, og maksimalt 3 fire spillere	Primært 25-32 år, snittalder på +/- 28 år	Følger A-lagets treningssyklus	Spiller A-lagets kamper
A-team players 10+1	Minimum 7 LUS, herav 2-3 KUS, og maksimalt 4 fri	Primært fordeling mellom 21-30 år, snittalder på +/- 26 år	Følger A-lagets treningssyklus	I tropp på A-lagets kamper. 2. lagskamp med mindre enn 45'
Utfordrere 3+1	Minimum 2-3 av 4 skal være KUS	U21 år	Følger A-lagets treningssyklus	2. lagskamp ved mindre enn 45'
Lærlinger 2	Spillere fra egen utviklingsavdeling	U19	Følger A-lagets treningssyklus m/tilpasninger	Spilelr primært på 2. laget og G19 NM
Hospitant 1 og 2 3-6	Spillere i egen utviklingsavdeling	U19	Følger 2. lagets treningssyklus m/tilpasninger	Spiller på 2. laget og G19 NM

Lik nåværende utgave og gjør samme funksjon. Denne utgaven hadde liten til ingen styling.

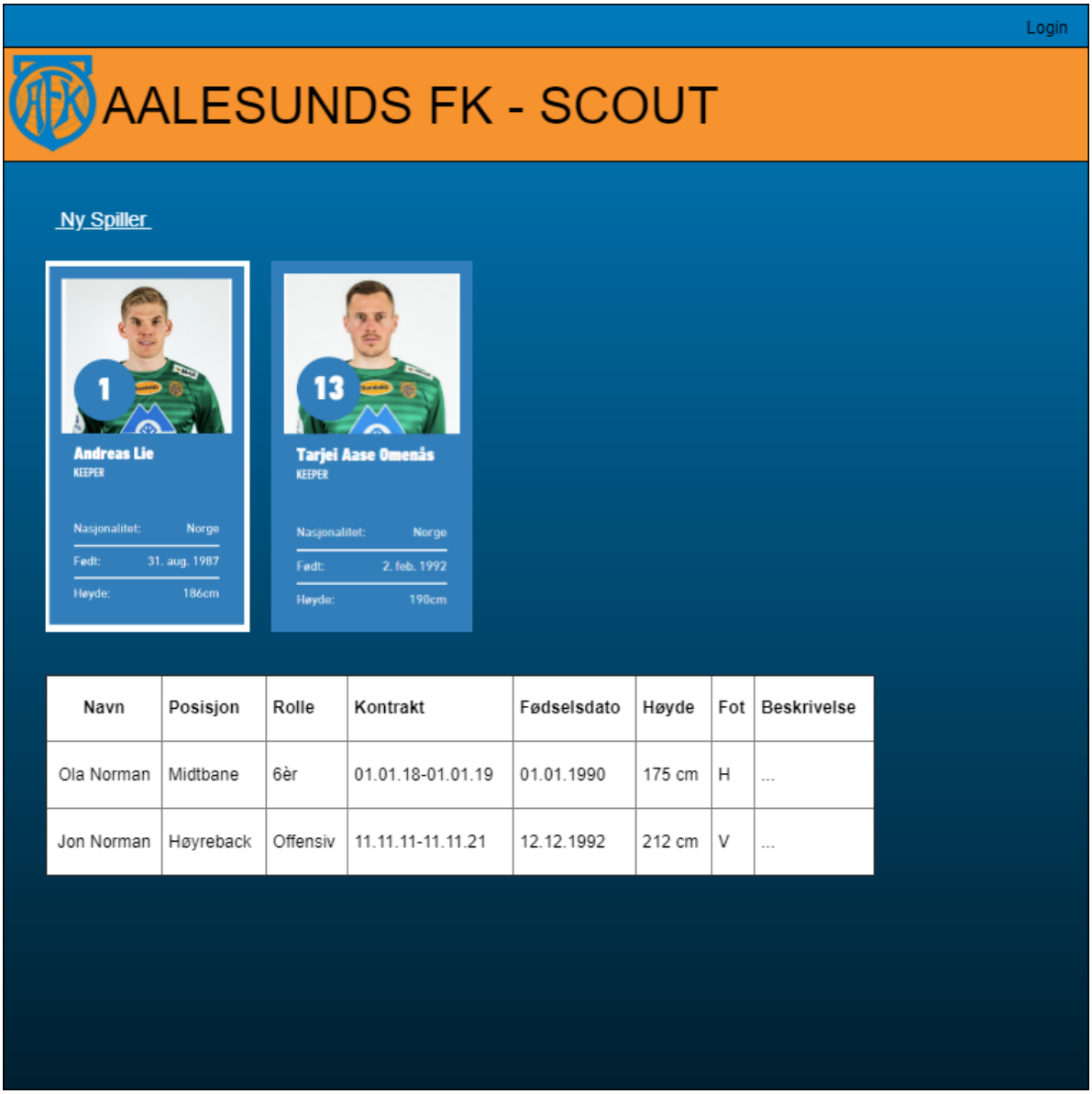
Prototype 5:
Spillerlisten.



Name	Height	Weight	Birth Date	Nationality	Preferred leg	Current Club	Former Clubs	Contract Expiry	Salary	Agent	Scout	Position	Description
Cristiano Ronaldo dos Santos Aveiro	187cm	86kg	Both	5. Feb 1985	Portugal	30.06.2022	Juventus FC	Forward - Left winger	Agent name	Placeholder	Placeholder	Placeholder	
Ronaldo de Assis Moreira	181cm	77kg	Right	21. Mar 1980	Brazil	-	Retired	Midfielder - Attacking midfield	Agent name	Placeholder	Placeholder	Placeholder	
Placeholder	Placeholder	Placeholder	Placeholder	Placeholder	Placeholder	Placeholder	Placeholder	Placeholder	Placeholder	Placeholder	Placeholder	Placeholder	
Placeholder	Placeholder	Placeholder	Placeholder	Placeholder	Placeholder	Placeholder	Placeholder	Placeholder	Placeholder	Placeholder	Placeholder	Placeholder	

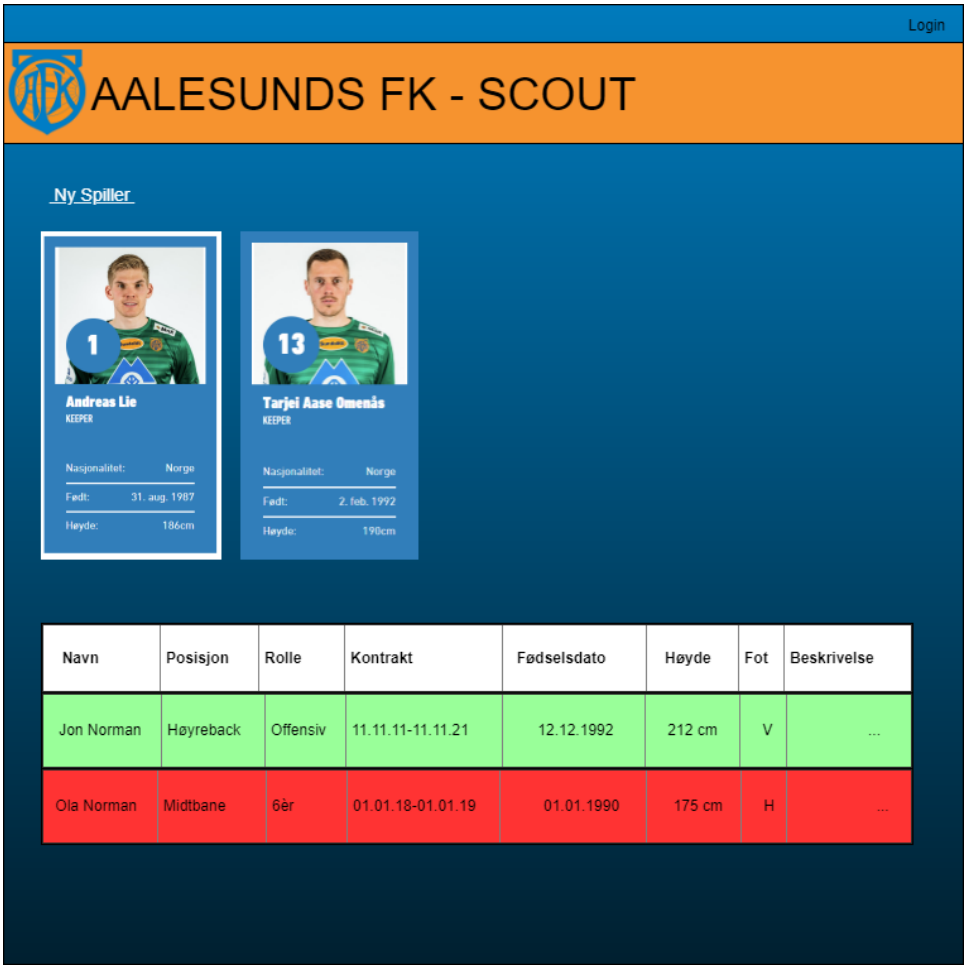
Denne siden tok oppdatert informasjon fra databasen på samme vis som den endelige utgaven av siden. Siden hadde en funksjon som bruktes til å sortere informasjon ved trykking på kolonnenavn. Denne utgaven hadde ingen ekstra søkefunksjon, ingen styling, alle elementer i listen ble lagt på samme side og siden viste mer informasjon enn det oppdragsgiver ønsket at skulle være synlig.

Prototype 6:
Konsepttegning for startside.



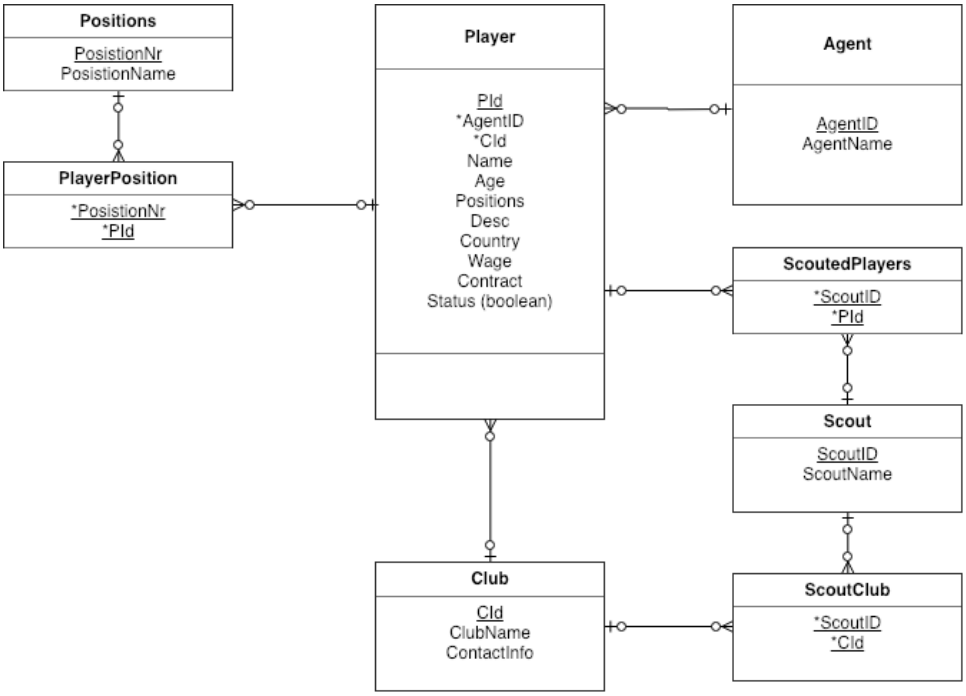
Første konsepttegning av startsiden til applikasjonen. Siden skulle ha en login/logut-knapp, vise alle favoritter øverst på siden, ha en liste over alle spillere og en knapp for manøvrering til en "Ny spiller"-side. Fargekoden var også en del annerledes, og har blitt endret etter anbefaling fra oppdragsgiver som ville ha en applikasjon som var mer lik AaFKs egen nettside.

Prototype 7:
Konsepttegning for startside med fargekode.



På denne konsepttegningen tok vi også med en fargekode for å kunne skille mellom spillere som var av stor eller liten interesse.

Prototype 8:
Første konsept for databasestruktur.



Denne modellen ble satt opp for planlegging av den aller første databasestrukturen. Denne strukturen er grunnlaget for strukturen i det endelige produktet og holder på mye av den samme informasjonen.